

SCALA

MULTIMEDIA MM200

English
manual

CUSTOMER SOFTWARE LICENSE

Please read this carefully before use!

You are purchasing a license to use SCALA(TM) Software. The Software is owned by and remains the property of SCALA and its licensor, is protected by international copyrights, and is transferred to the original purchaser and any subsequent owner of the Software media for their use only upon the license terms set forth below. Opening the packaging and/or using SCALA Software indicates your acceptance of these terms. If you do not agree to all of the terms and conditions, or if after use you are dissatisfied with your SCALA Software, return the unopened Software, manuals and any partial or whole copies within thirty (30) days of purchase to the party from whom you received it for a full refund.

GRANT OF LICENSE

SCALA AS ("SCALA") grants the original purchaser ("Licensee") the limited rights to possess and use the SCALA Software specified at the end of this License ("Software"), consisting of machine-readable computer code and documentation upon the terms and conditions specifically set out in this License.

TERM

This License is effective as of the time Licensee receives the Software, and shall continue in effect until Licensee ceases all use of the Software and returns or destroys all copies thereof, or until automatically terminated upon the failure of Licensee to comply with any of the terms of this License.

YOUR AGREEMENT

- * Licensee agrees that the Software will be used solely for Licensee's internal purposes, and that at any one time, the Software will be installed on a single computer only. If the Software is installed on a networked system, or on a computer connected to a file server or other system that physically allows shared access to the Software, Licensee agrees to provide technical or procedural methods to prevent use of the Software by more than one user at one time.
- * One machine-readable copy of the Software may be made for BACK-UP PURPOSES ONLY, and the copy shall display all proprietary notices, and be labeled externally to show that the back-up copy is the property of SCALA, and that its use is subject to this license. Documentation in whole or part may not be copied.
- * Use of the Software by any department, agency or other entity of the U.S. Federal Government is limited by the terms of the attached "Rider for U.S. Governmental Entity Users", which is incorporated by reference into this License.
- * Licensee may transfer its rights under this License, PROVIDED that the party to whom such rights are transferred agrees to the terms and conditions of this License, and written notice is provided to SCALA. Upon such transfer, Licensee must transfer or destroy all copies of the Software.
- * Except as expressly provided in this License, Licensee may not use, copy, disseminate, modify, distribute, sub-license, sell, rent, , lease, lend, give or in any other way transfer, by any means or in any medium, including telecommunications, the Software.

L I C E N C E A G R E E M E N T

LIMITED WARRANTY

SCALA warrants the Software media to be free of defects in workmanship for a period of ninety (90) days from purchase. During this period SCALA will replace at no cost any such media returned to SCALA, postage prepaid. This service is SCALA's sole liability under this warranty.

DISCLAIMER

License fees for the software do not include any consideration for assumption of risk by SCALA, and SCALA disclaims any and all liability for incidental or consequential damages arising out of the use or operation or inability to use the Software, or arising from the negligence of SCALA or its employees, officers, directors, consultants or dealers, even if any of these parties have been advised of the possibility of such damages. Furthermore, Licensee indemnifies and agrees to hold SCALA harmless from such claims. The entire risk as to the results and performance of the Software is assumed by the Licensee. The warranties expressed in this License are the only warranties made by SCALA, and are in lieu of all other warranties, expressed or implied, including but not limited to implied warranties of merchantability and of fitness for a particular purpose.

This warranty provides licensee specified legal rights, and licensee may also have other rights which vary from jurisdiction to jurisdiction. Some jurisdictions do not allow the exclusion of limitation of warranties, so the above limitations or exclusions may not apply.

GENERAL

This license is the complete and exclusive statement of the parties' agreement. Should any provision of this License be held to be invalid by any court of competent jurisdiction, that provision will be enforced to the maximum extent permissible, and the remainder of the License shall nonetheless remain in full force and effect. This License shall be controlled by the laws of the Commonwealth of Virginia and the United States of America, as applicable.

Sign and Mail the Registration Card Now, and Receive:

- * Notice of Software Upgrades
- * User Hotlines

SCALA Software Licensed:

Serial Number:

Version:

L I C E N C E A G R E E M E N T

RIDER FOR U.S. GOVERNMENTAL ENTITY USERS

This is a Rider to the SCALA AS. CUSTOMER SOFTWARE LICENSE, ("Licensee"), and shall take precedence over the License where a conflict occurs.

1. The Software was developed at private expense; no portion was developed with Government funds; is a trade secret of SCALA and its licensor for all purposes of the Freedom of Information Act, is "commercial computer software" subject to limited utilization as provided in any contract between the vendor and the Government entity; and in all respects is proprietary data belonging solely to SCALA and its licensor.

2. For units of the DOD, the Software is sold only with "Restricted Rights" as that term is defined in the DOD Supplement to DFAR 252.227-7013 (b)(3)(ii), and use, duplication or disclosure is subject to restrictions set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at 252.227-7013. Manufacturer: SCALA AS, Waldemar Thranesgt. 77, 0175 Oslo, Norway

3. If the Software was acquired under a GSA Schedule, the Government has agreed to refrain from changing copies of manuals or disks (except for backup purposes) and: (1) Title to and ownership of the Software and Documentation and any reproductions thereof shall remain with SCALA and its licensor; (2) use of the Software shall be limited to the facility for which it is acquired; and (3) if the use of the Software is discontinued at the original installation and the Government wishes to use it at another location, it may do so by giving prior notice to SCALA, specifying the new location site and class of computer.

4. Governmental personnel using the Software, other than under a DOD contract or GSA Schedule, are hereby on notice that use of the Software is subject to restrictions that are the same or similar to those specified above.

Scala, InfoChannel and the  logo are registered trademarks of Scala AS. All other products mentioned in this manual are trademarks of their respective owners. Copyright 1992 Scala AS. All rights reserved. No part of this publication, or any parts of this package, may be copied or distributed, transmitted, transcribed, recorded, photocopied, stored in a retrieval system, or translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, manual, or otherwise, or disclosed to third parties without the prior written permission of Scala AS.

T A B L E O F C O N T E N T S

Welcome to Scala MultiMedia!	1
Who are you?	2
Scala MultiMedia Quickstart	3
What is Scala MultiMedia?	4
What's new from version 1.13	6
System requirements	8
Package contents	8
Scala key	9
Installing Scala on your hard disk	10
What's in the Scala drawer?	12
What is a Scala script?	15
Starting Scala MultiMedia	16
 Tutorial	 17
Tutorial overview	18
A round trip	21
The number quiz	73
 Putting it all together	 102
What is multimedia?	103
The IFF standard	104
Pictures	106
Animations	112
Sound and music	116
 Reference	 118
Overview	119
General	121
The mouse	121
The keyboard	122
Common elements	123
Main menu	126
File menu	146
System menu	152
Edit menu	158

T A B L E O F C O N T E N T S

Font menu	168
Color palette	172
Layout menu	176
Text wipe menu	184
Page control menu	190
List menu	192
Button menu	196
Animation menu	202
Execute menu	206
Variables menu	208
Sound menu	212
Scala Lingo and ARexx	218
Overview	218
Commands	227
Scala EX	278
Overview	278
Sony Laserdisc EX	282
Pioneer Laserdisc EX	286
Canon ION	290
MIDI file player	294

Appendixes

- A Scala art library
- B Typefaces
- C Keyboard shortcuts
- D Utilities

Index



Welcome!

This chapter tells what Scala Multimedia is, and helps you off to a good start. Both new and experienced users will find important information here.

WELCOME TO SCALA MULTIMEDIA

Who are you?

Scala Multimedia is a flexible and powerful tool for multimedia presentations of every kind. For this reason, our users range from people with no deep technical knowledge of computers and software, to experienced computer "freaks" who know everything. This manual contains a lot of information, but not all of it applies to everybody.

- ◆ If you are new to Scala MultiMedia on the Amiga, we suggest that you read this *Welcome!* chapter first to get a basic idea about what Scala Multimedia is, and how to install the software. Then, move on to tutorial number one, for a round trip through the menus, with exercises that will teach you the basic functions of Scala Multimedia.
- ◆ If you are an experienced Amiga user, read the *Quickstart* section. This is quite likely all you need in order to create your first presentation. Then see the reference section or the advanced tutorials later on, for in-depth information about topics that interest you.
- ◆ If you are an experienced Scala user who has upgraded from Scala version 1.13, you don't need to go through the whole manual. See the *What's new* section on page 6 for information on new features.

WELCOME TO SCALA MULTIMEDIA

Scala Multimedia Quickstart

If you are familiar with the Amiga and the way it works, here is how to install and run Scala MultiMedia *now*. If you have never used the Amiga before, we suggest that you skip this and move on to the next section.

- ◆ Insert the *Scala key* in one of the joystick/mouse ports and turn on the computer.
- ◆ When the Workbench screen appears, insert the Scala MultiMedia main diskette.
- ◆ Double-click the *Install* icon on this diskette, and Scala Multimedia will be installed on your hard disk.
- ◆ When the installation is finished, open the Scala MultiMedia drawer on your hard disk, and then double-click the Scala MultiMedia program icon.
- ◆ The Scala Multimedia *Main* menu appears. Select *Load Script*, and load the script called *ScalaMainMenu.script* (located in the Scripts drawer). This is an interactive script that contains information and demos.
- ◆ Select *Run* in the *Main* menu. The script will start, and you can use either mouse button to select topics you want more information about.

A short guide to editing a presentation

- ◆ To make a new page, select *New* in the main menu and then select a background picture to use.
- ◆ When the *Edit* menu appears, type your text. When the page is finished, select *OK*.
- ◆ Select a pause and a wipe for the page in the *Main* menu.
- ◆ Make more pages in the same way, and view the whole script by selecting *Run* in the *Main* menu.

WELCOME TO SCALA MULTIMEDIA

What is Scala Multimedia?

Recently the word *multimedia* has become one of the most frequently used buzzwords in the computer industry. In most cases there has been more buzz than action, since the integration of pictures, video, sound, animation and user interaction requires lots of new technology. But the Commodore Amiga has always been prepared for these tasks, with excellent graphics, integrated four-channel stereo sound and easy integration with video equipment. The only thing missing has been software that makes it easy for you to mix all these elements together.

The Scala Multimedia system fills this need and brings all the exciting features of the Amiga to life.

What is it?

Scala Multimedia is a software package for creating on-screen presentations in a quick and elegant manner. It lets you put together presentations consisting of pictures, animations, text, sound and video with unparalleled ease. The package contains lots of ready-to-use backgrounds, fonts, symbols, sounds and music, created by leading Scandinavian designers to help you build attractive, professional presentations. When finished, your presentations can be run directly on the computer, or recorded to video.

Scala Multimedia is designed especially for non-technical users. Why make it harder?

How does it work?

The basic concept of Scala Multimedia is the *script*, which consists of a series of *pages*. You can add new pages in different ways, for example by loading a background picture and typing text on it. When a page is finished, it appears in the main menu. There you can decide how the pages should appear, how long they should be displayed, and if they are to be accompanied by sounds or music.

At any time, you can select *Run* to see how your presentation looks when the pages are shown in succession

WELCOME TO SCALA MULTIMEDIA

What can you do with Scala MultiMedia?

You can use Scala Multimedia to make anything from a simple, one-page title to add to a video, to an interactive map of lodgings, restaurants and attractions, displayed on a tourist information terminal.

- ◆ Add text overlays and symbols to your videos.
 - ◆ Use it as speaker support, instead of overhead projector or slides, when talking to an audience.
 - ◆ Use it as a stand alone presentation medium with "moving" advertisement and information to clients - for instance in a shop window.
 - ◆ Create an interactive training program, where the student answers questions and exercises, getting an immediate response.
 - ◆ Make advertisements and text information to be projected on a video wall, for instance at sports events.
 - ◆ Make animations, titles and other effects for TV programs.
 - ◆ Use it to format test results and technical information from other programs, to be presented in a visually attractive form.
 - ◆ Use it to present ideas to present ideas for advertising campaigns or proposed design programs.
- In a film project, use Scala for storyboarding.

To get an idea of some of the possibilities, run the demo scripts. These will show you how you can assemble pictures, sounds and animations into a breathtaking presentation, title your videos or make an interactive quiz.

WELCOME TO SCALA MULTIMEDIA

What's new in Scala Multimedia?

This section contains information for those who have upgraded from Scala 1.13. Here is a list showing the new features, and where to read more about them.

Scala EX

is a revolutionary new system of external program modules to control new types of input. Once an EX is installed, it becomes an integrated part of Scala. It is controlled from the main menu, through its own column. Scala Multimedia comes equipped with six EX modules. With no programming, you can incorporate laserdisk playback, Canon ION Still video, MIDI, CDTV sound and ARexx. New EXes will follow, to further expand the possibilities. This means that there are virtually no limits to what you can control from Scala.

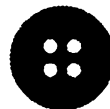
See page 278



Scala Buttons

make it even easier to set up your own hierarchies of interactive applications. New highlighting modes make it easier to choose the right button when using the script. You can start other scripts, ARexx commands and DOS applications, use variables and conditional branching. All this can be achieved with no programming or complex operations.

See page 196



Scala Shuffler

lets you see the whole presentation at a glance! The Scala Shuffler shows you miniatures of all pages you have made. You can shuffle them around with the mouse. You can choose from one to fourteen "slides" on one horizontal line, for up to 112 pictures on the screen at once!

See page 135



New wipes

have more than 25 amazing effects, such as stretch, flow, flip and cube - making Scala Multimedia "the software Toaster!". Crawl lets your messages fly horizontally across the screen. Out-wipes make text and brushes disappear, and the new *Link* function lets you select several lines to come on simultaneously.



WELCOME TO SCALA MULTIMEDIA



See page 218

LINGO

See page 206

Scala SnapLoad

A series of techniques developed to speed up loading and playback of pictures and animations. Preloading, dynamic and static buffering, and diskANIM are some of these techniques. With this feature you can play back animations of virtually any length, without having to install loads of memory.

Scala Lingo

is the "invisible" multimedia language that pulls the strings when you use Scala. If you need ultimate low-level control of every Scala function, use Lingo directly or from ARexx.

ARexx support

allows the scripts to be saved directly in ARexx language, and run as normal ARexx programs. In addition, Scala Buttons can launch ARexx scripts and use ARexx functions to communicate with any other software package that supports ARexx. These powerful features open up tremendous possibilities for the advanced users.

Record timing

A unique feature that synchronizes graphics with sound, video, MIDI etc. Instead of setting the time for each page (in the *Wait* menu), you simply record the timing of your presentation with mouse clicks. All timing functions in Scala have an internal accuracy of milliseconds.

Miscellaneous

- ◆ The main menu is now completely configurable. You can set the width of the columns by just clicking and dragging.
- ◆ Columns can be rearranged using the System menu.
- ◆ The new install utility can automatically scale the background pictures to fill the entire screen (overscan).
- ◆ Full support for sampled sounds and Soundtracker/DSS modules through the internal Sound EX

WELCOME TO SCALA MULTIMEDIA

System requirements

In order to run Scala Multimedia you need a Commodore Amiga with Kickstart version 1.3 or newer, 1MB of CHIP memory, 2 MB fast memory and a hard disk. It is possible to run the program with less memory and without a hard disk, but this is not recommended and may cause you a lot of problems. The more memory Scala Multimedia gets, the better the performance. Processors like 68020 and 68030 will considerably speed animation playback.

To use all the powerful functions of the Scala Multimedia system, additional equipment like Laser Disk players, video equipment, genlocks and color or black/white printers may be obtained.

Scala Multimedia is compatible with a wide range of Amiga software. Products for painting and animation (like Deluxe Paint IV from Electronic Arts), image manipulation (like Art Department Professional from ASDG) and sound editing (like Digital Sound Studio from GVP) can enhance your presentations.

Package contents

Scala Multimedia comes on 8 diskettes, These diskettes contain:

- ◆ ScalaMM program, ScalaMMPlayer and ScalaMMPrint
- ◆ Different fonts with varying sizes and weights.
- ◆ 59 high-quality background pictures
- ◆ 70 symbols
- ◆ Demo scripts that shows examples of video titling and interactive scripts
- ◆ ..and more.

In addition there is the Scala Key, a keyword reference card and, of course, this manual. You will also find a user registration card which we request you to fill out and mail immediately, in order to receive info on new versions and upgrades. To receive telephone support, you need to be registered with us.

The Scala Key

This is small device that must be present in one of the mouse/joystick ports when you run Scala Multimedia.



Warning: It is best to turn the Amiga off before inserting the key. Otherwise, you may blow an automatic fuse in the Amiga. If this should happen, simply turn the Amiga back on.

The key has a pass-through 9-pin connector, so the port is not blocked if you need it to connect a joystick or mouse. The only restriction is that you can not connect a device that uses the joystick port to send signals *out* of the computer, but this mainly applies to a small number of genlocks and RGB video splitters. These devices are connected to port #2, and to avoid problems you have to put the key in port #1, where the mouse is usually connected.

If the key is not present, a message will appear, prompting you to put the Scala Key in one of the mouse/joystick ports. Insert the key and select OK. If the key is already present, check to see that it is properly inserted.

You can move the key from one computer to another, but it must always be present for ScalaMM to run. If you want to run ScalaMM on more than one computer simultaneously, you will need to purchase an additional program copy for each computer.

WELCOME TO SCALA MULTIMEDIA

Installing Scala on your hard disk

Before you can start using Scala Multimedia you have to install the software on your hard disk. The disk labeled "Disk 1" contains an installation program that will perform this operation for you. Turn on your Amiga in the usual way. When the Workbench screen appears, insert the Scala Multimedia program disk. Click the diskette icon twice with the left mouse button. The disk window opens, and you can then double-click the icon marked "Install".

The install program will guide you through the installation. Follow the instructions on the screen.

The Scala fonts will be installed in the FONTS: drawer. If your hard disk is divided in two or more partitions, and the FONTS: drawer is located on your SYS: partition, there may not be enough free space on this partition. If there is not room for the fonts, you will have to install them on another partition of your disk, and then use the *assign* command to tell Scala MultiMedia to look for the fonts there.

WELCOME TO SCALA MULTIMEDIA

Here is how to tell Scala to install the fonts on your Work: partition. (If you use an old Commodore A2090 controller, look below).

- 1) Open a CLI or Shell. Type: "mkdir work:fonts"
- 2) Type "ED s:startupII" and press RETURN. (If you use Workbench 2.0, type "ED s:user-startup" instead.
- 3) Add the line "assign fonts: work:fonts"
- 4) Press ESCape, "X" and then RETURN
- 5) Reboot the machine (Control-Amiga-Amiga)
- 6) Run *Installl* again.

If you use an Amiga 2000 with an "old" A2090 hard disk controller:

- 1) Open a CLI or Shell. Type: "mkdir work:fonts"
- 2) Type "ED boot:s/startup-sequence"
- 3) Change the line with "assign fonts: sys:fonts" to "assign fonts: work:fonts"
- 4) Press ESCape, "X" and then RETURN.
- 5) Reboot the machine (Control-Amiga-Amiga)
- 6) Run *Installl* again.

WELCOME TO SCALA MULTIMEDIA

What's in the Scala drawer?

If you performed a complete installation, the following files and drawers will appear in the Scala Multimedia drawer on your hard disk. If you choose *not* to install files that are not necessary in order to run Scala Multimedia such as the demo files), you may not find all the files mentioned here.

Files

ScalaMM	This is the main program, and it is <i>this</i> icon you double-click to start Scala Multimedia.
ScalaMMPlayer	This is a program that can be run to play back the scripts. ScalaMM calls this program when you select <i>Run</i> in the <i>Main</i> menu. The ScalaMMPlayer will run independently when you double-click a script icon on the Workbench. The script will be loaded and run directly.
ScalaMMPrint	This is a utility that lets you make two kinds of hard copies of your scripts, either as pictures or as text. See the reference section for a full description of ScalaMMPrint.

WELCOME TO SCALA MULTIMEDIA

Drawers

Scripts	This drawer contains demo scripts, and you can use it to save your own scripts.
Backgrounds	Here you will find the professionally designed background images that are included in the Scala Multimedia package.
Animations	This drawer contains an animation used in the tutorial, plus the demo animations (if installed).
Symbols	Here you will find a range of symbols that can be loaded into your scripts using the <i>Load</i> button in the <i>Edit</i> menu.
Palettes	In the Edit menu you can load a new color set (<i>Load Palette</i>).
Sounds	This drawer contains a range of IFF sampled sounds that you can use from the Scala Multimedia sound menu.
Music	A handful of different melodies in the Soundtracker Module format. These can also be accessed from the <i>Sound</i> menu.
Layouts	When you save a layout, put it in this drawer. This drawer contains the layouts that are used in the tutorials.

WELCOME TO SCALA MULTIMEDIA

ARexx Scala Multimedia has full support for the ARexx programming language that is often used to exchange information between different applications. In the ARexx drawer you will find examples for Scala Multimedia. Full documentation is found in the ARexx/Lingo chapter.

EX After you install Scala Multimedia, this drawer will contain all the bundled EX'es.

Startup When you start Scala Multimedia, all programs contained in this drawer will also start. For example, all EX modules that you use should be moved to this drawer. If they are not located in the Startup drawer, they will not appear in the menu. For example, you can drop the Deluxe Paint icon in the startup drawer if you want it to be available to make new backgrounds and symbols when you run Scala Multimedia.

Utilities This drawer contains some extra utilities for animation manipulation etc. See the *Utilities* appendix.

Some hidden files

The installer utility will place a file called ScalaMM.library in the LIBS: directory, and a ScalaMM.device in the DEVS: directory on the system partition of your hard disk. In addition, there is a Scala font in your FONTS: directory, in three point sizes (6, 8 and 11).

What is a Scala script?

Before you start working, use a few minutes to read this chapter in order to better understand how Scala Multimedia looks at scripts, pictures, animations and sounds.

The term *script* that is used in this manual, is really nothing more than a detailed list of all the events in your presentation; that is, which picture to load, where to put the text, in which font, what wipes to use, timing, sound variables, animation playback speed etc. When you run the script, Scala Multimedia executes the instructions, and loads the pictures, animations and sounds as they are needed. This means that a script containing hundreds of events will still have a moderate size, since nothing is stored twice. It also makes it easy to modify sounds, pictures and animations from outside Scala Multimedia, since the new version of the file will be loaded the next time you run the script.

Keep your scripts running

While this approach saves space and time, it requires some care on your part. The script contains information about where files are located; the name of the hard disk and the directory. This means that if you start deleting, moving and renaming files without knowing if they are part of a script, you may run into problems. Scala Multimedia will not be able to find the file, and an error message will be displayed.

A word of advice is to organize your files in the following way: Make a new drawer (using the Workbench), and move all the pictures, animations and sounds that are used in a script into this drawer. If your script only contains standard Scala backgrounds and sounds, this is not necessary. But if most of your script is a large project with loads of "custom made" backgrounds, animations and sounds, this technique can make things a bit easier.

The *FixScript* utility can find locate missing files. It will scan through the available disks, searching for files with the same names as the missing ones, and then update your script to use these files instead. See the *Utilities* appendix.

WELCOME TO SCALA MULTIMEDIA

Starting Scala Multimedia

When you have installed the software and the Scala Key, you are ready to start working. Boot your Amiga in the usual way. When the Workbench screen appears, double-click (press the left mouse button two times) on the icon of the hard disk that contains the Scala Multimedia drawer. Then double-click on this drawer, and finally double-click on the Scala Multimedia program icon. Within a few seconds the main menu will appear.

If the program cannot to start, it will present an error message explaining what went wrong. Problems are mainly caused by missing keys, improper installation or insufficient memory. In the latter case, if you know that you have at least two megabytes of RAM, check to see if any other applications are running (and close them).



You are now ready to start using the program. Scala Multimedia is very simple to use despite of its powerful features. Therefore, you could probably just close this manual and start experimenting on your own. However, you might miss a few great features, so we suggest that you begin with the tutorial chapter.

If problems occur while you are working, use the reference section to get information about a specific topic.



Tutorial

This is the easiest way to learn how to use Scala MultiMedia. Here are lessons that take you step-by-step through the menus and features of Scala MultiMedia.

TUTORIAL: OVERVIEW

A tutorial overview

This tutorial section is divided in two parts. The first one will show you how the most important functions work. The other one shows you how to make an interactive quiz game.

Tutorial one: A round trip

This tutorial is recommended for all users. We lead you step-by-step through the basic functions of Scala MultiMedia:

- ◆ Adding and editing new pages
- ◆ Loading animations
- ◆ Selecting page and text wipes
- ◆ Using sound

Tutorial two: An interactive presentation

The *Scala Buttons* found in Scala MultiMedia herald a new era in interactive multimedia. In tutorial two we will show you how to build an interactive application with buttons and variables. The example is a quiz game where the user gets response and scores as the game proceeds.

TUTORIAL: OVERVIEW

Conventions used in the tutorial

To make it easier for you to understand the tutorial, here is an explanation of some of the most frequently used words.

Select	This means to move the pointer inside a button and click the left mouse button once.
Double-click	This means to click the left mouse button quickly two times.
Drag	When you drag something (a text or a slider), you move the pointer to it, press the left mouse button, and then move the pointer while holding the mouse button down. Release the mouse button to stop dragging.
Cursor	When you use the <i>Edit</i> menu, there is a square on the page that shows where the next letter you write will appear. The cursor also indicates the <i>current</i> line. It can be moved about using the cursor keys or the mouse.
Pointer	This is the small arrow on the screen that moves when you move the mouse.
Script	The list you see in the <i>Main</i> menu is the visible part of the script. It contains all the information about which backgrounds, fonts, colors, wipes etc. that the presentation is made of. When you make new pages, they are added to the script. When you save a script, it is saved to a file which contains all this information.

TUTORIAL: OVERVIEW

Page	This is what make up a line in the script. Usually it is a picture or an animation.
Menus & buttons	These names are written in italics throghout this manual. Examples are the <i>Main</i> menu and the <i>OK</i> button.
Do this...	Whenever you are supposed to do something, this is indicated by a square, just like this: ☐ First do this ☐ Then this ☐ And finally this.

Illustrations

On every page in this section of the manual there is an illustration that shows how the screen should look like when you have reached that point in the tutorial.

Take a break

If you need to interrupt your session with the tutorial, remember to save your work before you quit Scala MultiMedia. This is described on page 37.

Good luck!

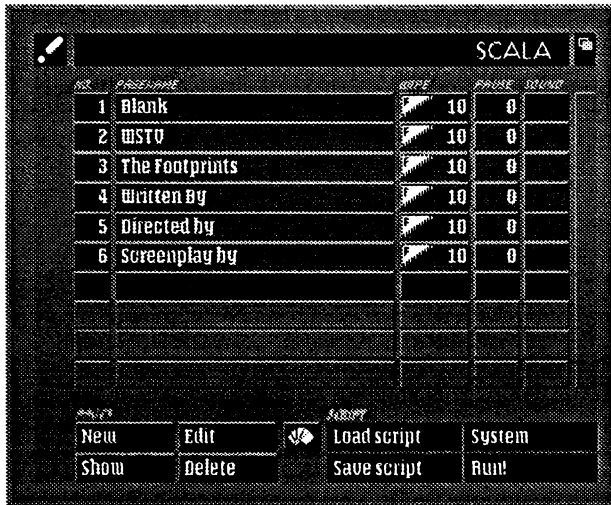
Tutorial one - a round trip

In this tutorial we are going to make a script containing several pictures, an animation, sound and music. After completing this lesson you will know how to use the basic functions of Scala MultiMedia. You will be able to :

- Load a picture*
- Write and edit text*
- Change color and style of the text*
- Move text*
- Save the script*
- Select transitions and speed*
- Use the Scala Shuffler*
- Use the sound EX*
- Load an animation*

To run this tutorial, Scala MultiMedia and the demo files must be installed on your hard disk.

TUTORIAL: ROUND TRIP



The Main menu

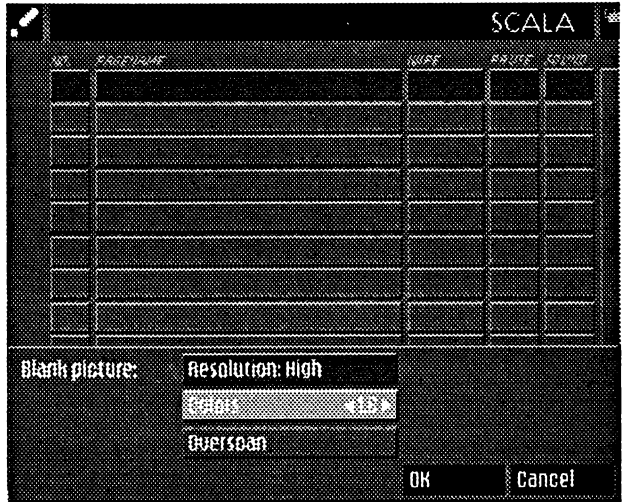
Start Scala MultiMedia as described in Chapter one. The *Main* menu appears. This menu contains a list of the pages in the script, and the buttons controlling scripts and pictures.

There are columns of buttons for the page numbers, page names, wipes, pauses, sounds and other EX modules.

The script is displayed from top to bottom, where the settings on the same horizontal line "belong" to each other. This means that the wipe in the wipe column tells how the page on the same line is supposed to appear.

If the script exceeds 10 pages, use the slider to the right of the script list to move up and down.

TUTORIAL: A ROUND TRIP



Making a blank page

The first page in the script will be white text on a black background. Now we are going to create a black background page.

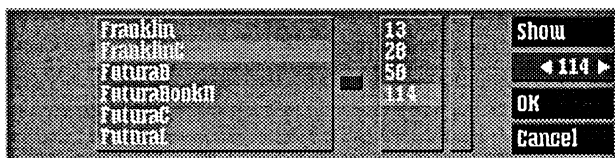
- ☐ Select *New*. The *File* menu appears.
- ☐ Select *OK* without selecting a file name first. The *Blank picture* menu appears (shown above).
- ☐ Press the buttons in the menu to cycle through the available options. Select *Resolution: High*, *Colors: 4*, *Overscan: None*.
- ☐ Select *OK*.

TUTORIAL: A ROUND TRIP

Change font

Now we are going to change the typeface that your name is written in. On the *Text* menu there is a *Font* button. It shows the name and size of the font where the cursor is. This button is also used to activate the Font menu.

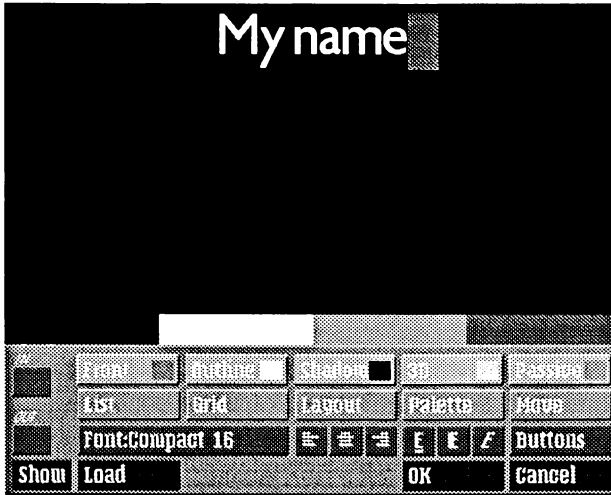
- ☐ Select the *Font* button. The *Font* menu appears.



Here you can select a typeface. The available fonts are in the list to the left, and the sizes of the font are given to the right. The font list may contain more fonts than it is possible to view at once. To look through the list, use the slider to the right. The same principle is used in the font size list. To preview the selected font and size, you can click the *Show* button. A line of letters and numbers will then be displayed on the screen. Click the *Show* button once more to deactivate the function.

- ☐ Select GillN.
- ☐ Select the size 58 in the size list.
- ☐ Select *Show* if you want to see what the font looks like.
- ☐ Select OK.

TUTORIAL: A ROUND TRIP



Automatic Justification of the Text

Scala MultiMedia has three ways to align or justify text: center, left and right. You will find these buttons just below the *Layout* button.

To center a line, simply select the center justification button. To disable centering, select the button once more. If you select another kind of justification for a line, the new justification will replace the old one, because only one can be used at once.

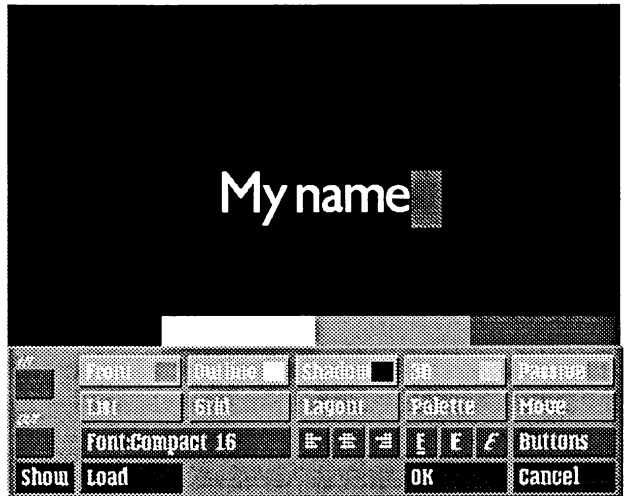
Now we are going to centre your name.

- ☐ Check that the cursor is in your name.
- ☐ Turn on the centre justification. You will see that your name moves exactly to the middle of the screen horizontally.
- ☐ Try to move your name, you will find that you can move it up and down - but it is locked sideways!



The justification symbols of Scala: left, center and right justification.

TUTORIAL: A ROUND TRIP



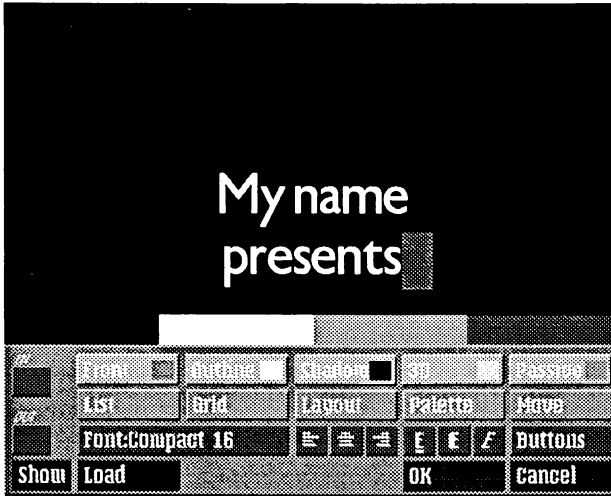
Moving

Your name is now at the top of the screen, and the cursor is right after the last letter in your name. Now we are going to move the text to a place just above the middle of the screen.

On the right side you will find the *Move* button. When this button is selected, Scala MultiMedia enters *Move* mode, which means that the menu disappears, and you can use the mouse to drag text and brushes on the screen. When you click the right mouse button, the *Edit* menu appears again, and you exit *Move* mode.

- ☐ Select the *Move* button. The menu disappears and you can see the whole page.
- ☐ Move the pointer to your name.
- ☐ Drag your name so that it is just above the center of the screen.
- ☐ Release the left mouse button, and click the right mouse button to get back the *Edit* menu.

TUTORIAL: A ROUND TRIP



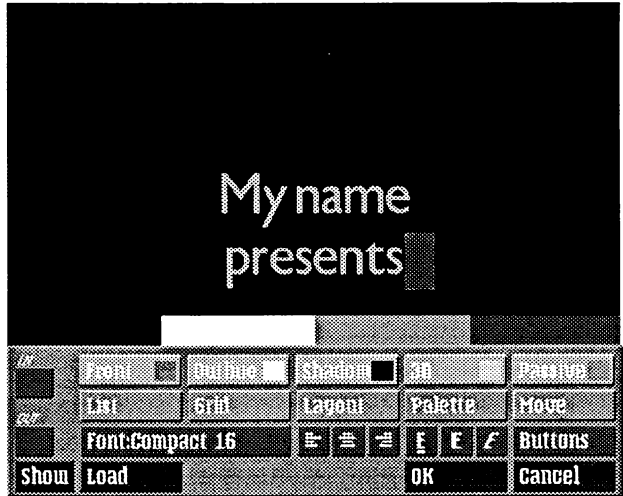
Write Another Line

The cursor is still to the right of your name. Press the RETURN key (the large key with an angled arrow), and the cursor will jump to the next line. Since centering is still turned on, the text will appear in the middle of the screen.

- Type: "presents"

You now have two lines of text on the screen. To move the cursor from one line to another, use the up/down arrow keys.

TUTORIAL: A ROUND TRIP



Alter Color

You can have a different color for each line of text. Now we want to change the color of the two lines you have written to gray. The color bar above the *Edit* menu is used to set the colors on the letters, shadows, outlines etc. When you select a color, a border appears around the color. To copy this color to the current text line, click inside the small square on the *Front* button. The color of the square will change to the selected color, and so will the color of the current line.

- ☐ Select light gray in the color bar..
- ☐ Click the color square on the *Front* button.
- ☐ Press the cursor down key to move the cursor to the line.
- ☐ Click the color square on the *Front* button.

TUTORIAL: A ROUND TRIP



Save the picture

To save the picture, select *OK* in the *Edit* menu. You will then be asked what the page is to be called. Scala MultiMedia will suggest the first line of the text as the page name. If you want to use another name, simply type another name inside the field.

- ☐ Select *OK*.
- ☐ The *Name page* menu appears. If it is OK to use your name as the page name, select *OK*.

The *Main* menu appears, and there is one page in the script list.

TUTORIAL: A ROUND TRIP



Edit the page

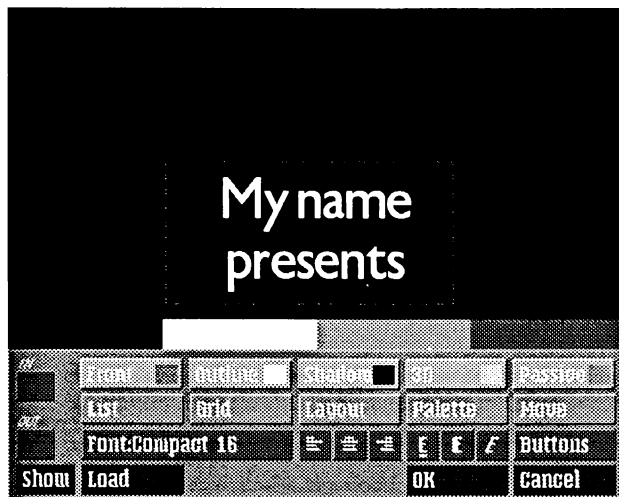
We want to change the text colors on the first page back to white. First we have to go to the page. When you see the *Main* menu, you notice that the page with your name is written in yellow (highlighted). Regular page names are written in white, but one line is always written in yellow. This is the *current* page in the script, and it is the last page you selected or edited.

When you select a page command, it will apply to the current page. When you delete pages, the current page is the first to be deleted. When you show pages, the current page is the first one to be shown, and when you select *New*, the new page will always be inserted after the current page. To edit a page, just select it and then select *Edit*. Now we are going to edit the page that has your name:

- ☐ Select *Edit*.
- ☐ When the *Blank picture* menu appears, select *OK*. The current page appears on the screen.

.

TUTORIAL: A ROUND TRIP

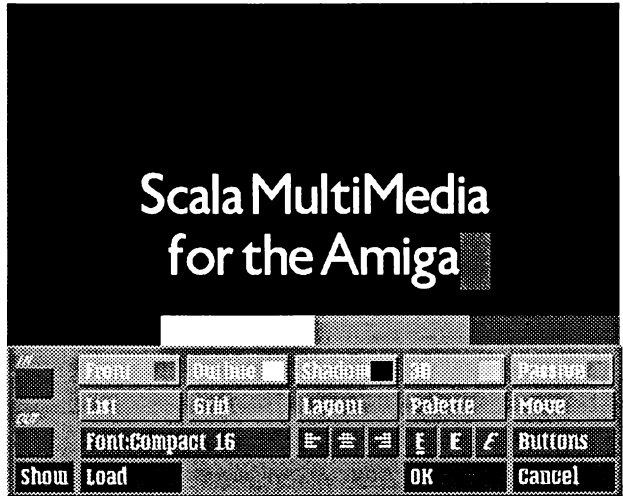


Multiple selection

When we changed the color of the lines to light gray, we changed one line at the time. Scala MultiMedia lets you work with more than one line simultaneously by using the "rubber band". When you click somewhere on the page and then hold the mouse button down while you move the pointer, a dotted square will appear as you drag. When you release the mouse button, the square will freeze.

The square selects all objects that it touches or encloses. They will be affected by your next command

- ☐ Make a rubber band around the two text lines.
- ☐ Select the white square in the color bar.
- ☐ Click the color square on the *Front* button. Both lines turn white.

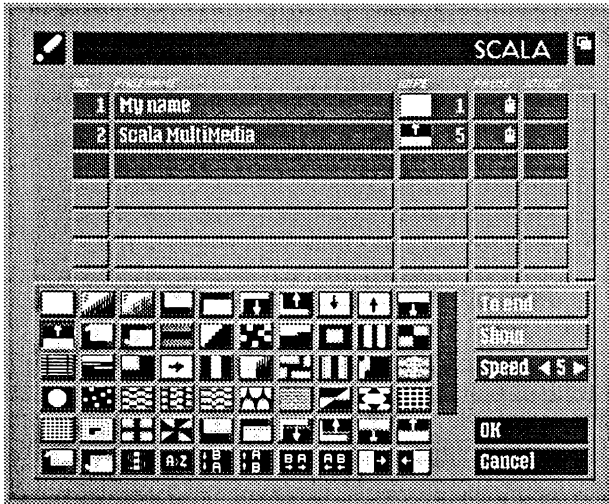


Make a page with the same layout

We want to make a page quite similar to the first one. When you are saving a page, there is a button called *Save and new* that saves the page and creates a new one with the same background and the same text lines. The text lines are empty - you just need to fill them!

- ☐ Select OK, and the *Name page* menu appears.
- ☐ Select the *Save and new* button to make a new page using the same layout. The *Edit* menu appears again, but with no text. The cursor is on the first of the two lines.
- ☐ Type "Scala MultiMedia" and press RETURN.
- ☐ Type "for the Amiga".
- ☐ Select OK and the *Name page* menu appears.
- ☐ Select OK in the *Name page* menu, and you are back in the *Main* menu.

TUTORIAL: A ROUND TRIP



Run the script

Once back to the *Main* menu you can try view what you have made.

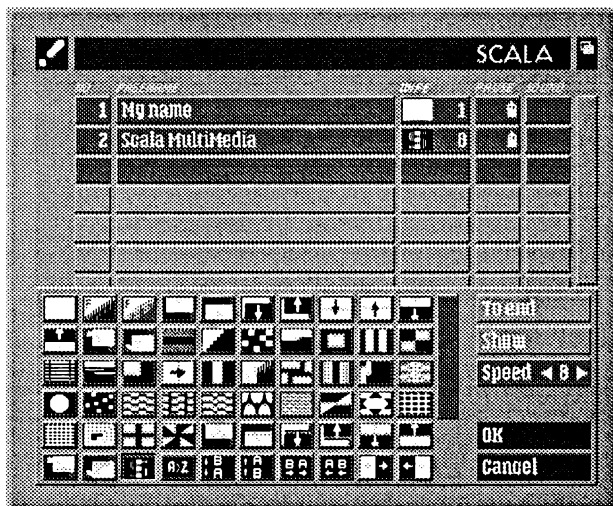
- ☐ Select *Run!* to the bottom right. Page number one will appear first.
- ☐ To see the second page, click the right mouse button.
- ☐ Use the *ESCAPE* key to return to the *Main* menu.

The transition between pages was dull. How about making the second page push the first one up?

- ☐ Select the *Wipe* button on the second line in the script. The *Wipe* menu appears.
- ☐ Select the wipe that looks like the one to the right.
- ☐ Select *OK*.
- ☐ Run the script again.



TUTORIAL: A ROUND TRIP

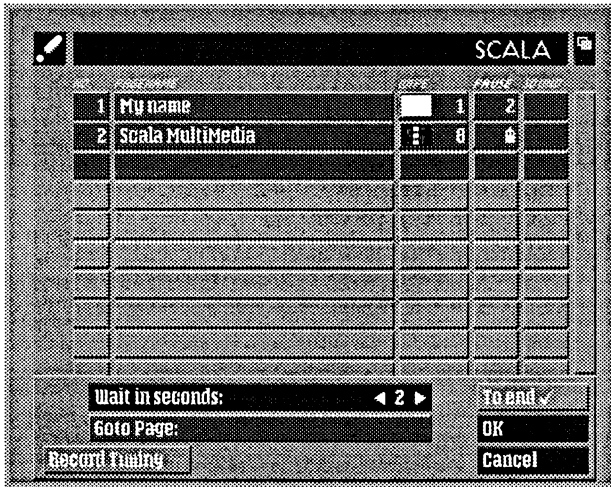


You will see that when you click the right mouse button, the second page pushes the first one up. How about trying a different wipe?

- ☐ Select the *Wipe* button for the second script line, and the *Wipe* menu appears again.
- ☐ Select any wipe!
- ☐ Select *Show* to see what the wipe looks like. Use the *ESCAPE* key to return to the *Main* menu.
- ☐ Repeat the previous steps if you want to see more wipes.
- ☐ Select the "superimpose" wipe, which has a button that looks like the one to the left.
- ☐ Change the speed of the wipe to 8 by using the right arrow on the *Speed* button.
- ☐ Select *OK*, and run the script.



TUTORIAL: A ROUND TRIP

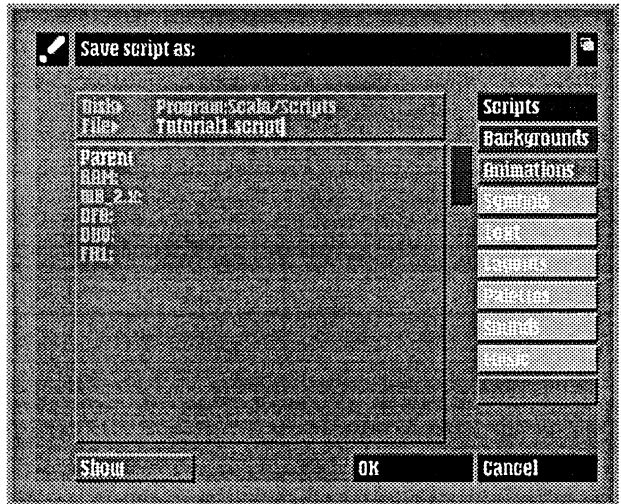


Setting the pauses

If you want the pages to advance without waiting for a mouse click, set the timing with the *Pause* buttons.

- ☐ Select the *Pause* button for page number one. The *Pause* menu appears.
- ☐ Click the increase arrow (pointing right) on the *Pause* button three times, so that the value is set to two seconds.
- ☐ We want the next page in the script to have the same pause, so select the *To end* button (a tickmark appears).
- ☐ Select OK.
- ☐ Run the script. Notice that the pages are displayed for two seconds.

TUTORIAL: A ROUND TRIP

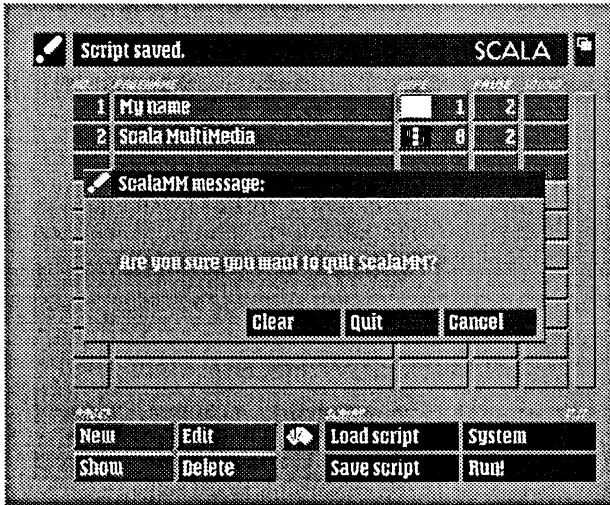


Save the script

It is always smart to save your work while working, in case something goes wrong. As you make the pages, they are *not* saved on your hard disk, just in the memory. So if a power failure should occur, all pages not saved to disk will be lost. Saving the script to disk is done in the *Main* menu, with the *Save script* button.

- ☐ Select *Save Script* from the *Main* menu. The *File* menu appear, waiting for you to select where to save your script and what to call it.
- ☐ To save the script in the drawer called "Scripts", click on the *Scripts* button to the right. The contents of the drawer are shown on the screen.
- ☐ Now you can type a name in the *File* field. Use the name *Tutorial1.script*.
- ☐ Click on the *OK* button at the lower right of the screen. When the script is saved, you return to the *Main* menu.

TUTORIAL: A ROUND TRIP



How to Quit Scala MultiMedia or clear the script.

If you don't want to continue with this tutorial now, or if you want to clear the script, select the exclamation mark (the Scala logo) in the top left corner of the screen. Then you will be asked if you want to quit Scala MultiMedia, clear the script, or return to the *Main* menu. Use the *Clear script* option when you want to remove all the pages in the script, to start working with a new one.

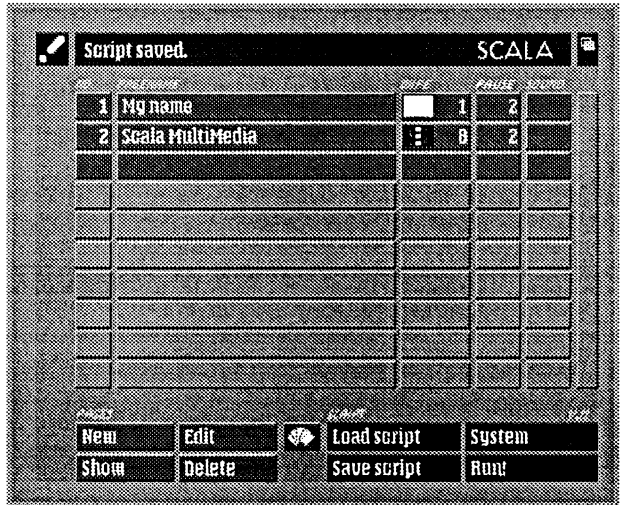


The next thing you will learn in this tutorial is to load a script, so if you don't want to quit now, we'll clear the script anyway. Remember that anything not saved to disk will be lost when you quit Scala MultiMedia or clear the script.

This is how you clear the script:

- ☐ Select the exclamation mark. The *Quit Scala MultiMedia* menu appears.
- ☐ Select *Clear*, and you are back in the *Main* menu, with no pages in the script.

TUTORIAL: A ROUND TRIP

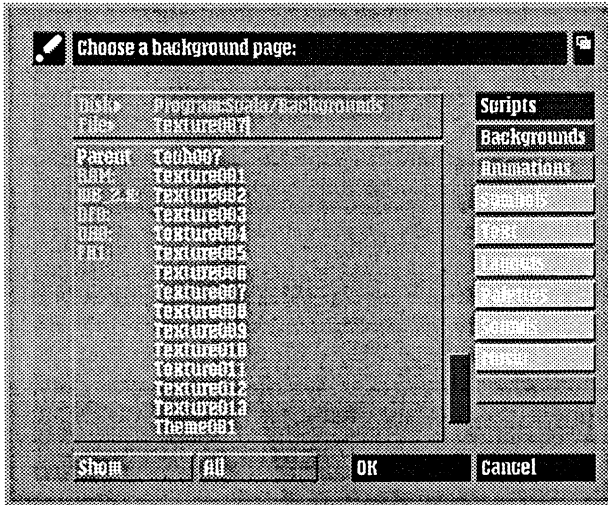


Loading the script

If it is not already running, start Scala MultiMedia in the usual way. Now we are going to load the script you saved earlier.

- ☐ Select *Load Script* from the *Main* menu. The *File* menu appears.
- ☐ Click the *Scripts* button to see the contents of the "Scripts" drawer.
- ☐ Select the script called "Tutorial1.script".
- ☐ Select *OK*.

TUTORIAL: A ROUND TRIP



Loading a background picture

The first two pages we made had blank, single-color backgrounds. On the next page we want to have a background picture.

- ☐ Select *New* in the *Main* menu to load a background picture.
- ☐ Click the *Backgrounds* button to enter the drawer where the Scala MultiMedia backgrounds are located.
- ☐ Use the slider on the right side of the file list to bring the background called "Texture007" into view.
- ☐ Select it (the color of the name changes from white to yellow).
- ☐ Select *OK*.



Typing more text

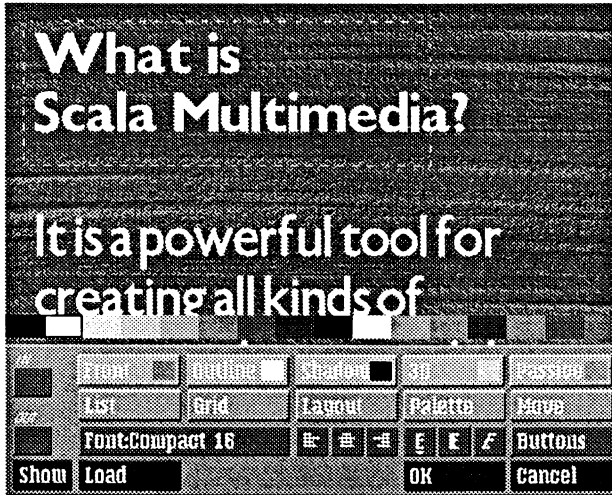
This page will contain more text than the previous ones. Let's start typing!

- ☐ Use the *Font* menu to select GillN 58, just like before.
- ☐ Select left justification.
- ☐ Type the following, pressing the *RETURN* key after each line. Use the *RETURN* key also to make a blank line after the heading.

What is
Scala MultiMedia?

It is a powerful tool for
creating all kinds of
presentations containing
pictures, sound and video.

- ☐ Click the right mouse button to get the *Edit* menu back.



Changing the font

We want the two first lines to be written in another font, in order to emphasize that they are a headline.

- ☐ Use the "rubber band" to select the two first lines.
- ☐ Select the *Font* button to bring up the *Font* menu.
- ☐ Select *Gill* in the font list.
- ☐ Select 57 in the size list.
- ☐ Select OK.

The first two lines will now change to the typeface Gill 58, which is a bit heavier than GillN.



Changing the color

We want the heading to have yellow letters instead of white ones. Since we previously have used the rubberband to select the two first lines on the page, we can change the colors as well while the rubberband is still there.

- ☐ While the rubberband still covers the two first lines, click the yellow color in the color bar.
- ☐ Click the color square on the *Front* button.
- ☐ The color of the heading changes to yellow.

What is Scala Multimedia?

It is a powerful tool for
creating all kinds of
presentations containing
pictures, sound and video.

Adding a shadow

To make the letters more visible, and add more depth to the picture, we want to add a drop shadow on all the text lines on the screen.

- ☐ Use the rubberband to select all the text lines on the screen. Use the right mouse button if you need to turn the *Edit* menu off or on.
- ☐ Select the *Shadow* button. Don't click inside the color square, but on the button itself.

All the text lines on the screen get a black drop shadow.

If you wanted another color on the shadow, you could change it by selecting the desired color in the color bar, and then clicking the color square on the *Shadow* button. But remember that the shadow of the color looks best if it is black or a darker version of the background color. In this case, black looks best.

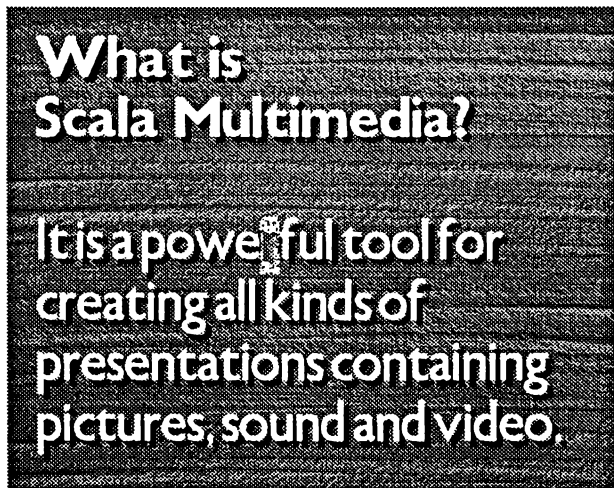
What is Scala Multimedia?

It is a powerful tool for
creating all kinds of
presentations containing
pictures, sound and video.

Modifying the shadow length

In the *Layout* menu, you can change many settings controlling text appearance. Details about shadows, outlines, line spacing etc. can be modified in this menu. Now we are going to increase the distance between the text and the shadow, to "raise" the letters from the background even more.

- ☐ With the rubberband still covering the whole screen, select the *Layout* button to make the *Layout* menu appear.
- ☐ Use the slider to make the *Shadow length* button appear in the button list in the middle of the menu.
- ☐ Use the *increase* arrow to set the value to six.
- ☐ Select *OK*.



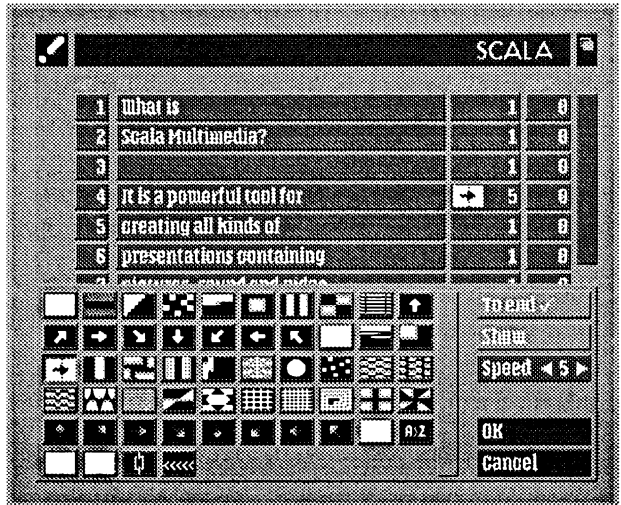
Text wipes

Not only the pages, but also the text lines on a page can enter the screen in a number of different ways. This is ideal for directing the viewers' eyes, and makes it easier to present a long message. Text and objects can enter and leave the screen in lots of different ways.

In this script, we want the text lines to appear from left to right, as if they are typed.

- ☐ Select the line "It is a powerful tool for".
- ☐ Select the *IN* button to the left. The *Text wipe* menu will appear.
- ☐ Select the wipe that has an arrow pointing right.
- ☐ Select *To end* to make all the following lines use the same wipe, and then select *OK*.

TUTORIAL: ROUND TRIP



Using the *List* menu

Many operations on text lines can be done easily in the *List* menu. In this menu you can handle text lines just like you handle pages in the *Main* menu.

- ☐ Select *List* from the *Edit* menu. The *List* menu appears.
- ☐ Select the *Wipe* symbol for the line "It is a powerful tool for".
- ☐ In the *Text wipe* menu, select the button that looks like the one to the left.
- ☐ Change the speed to 8.
- ☐ Turn on *To end*.
- ☐ Select *OK* in the *Text wipe* menu.
- ☐ Select *OK* in the *List* menu, and you're back in the *Edit* menu.





Making more pages

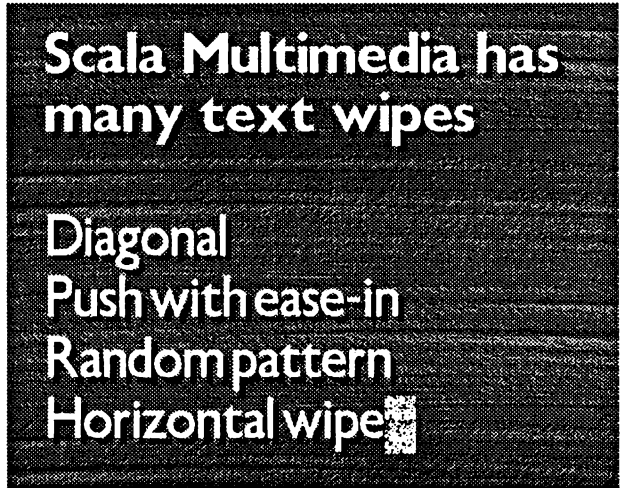
Remember the easy "Save and new" option we used earlier? There is an even easier way to create a new, similar-looking page. If you press the F3 function key, a new page with the same background and layout will be created.

- ☐ Press the F3 function key. The page is saved, and a new one is created.
- ☐ Type the following text

A complete
system

Scala MultiMedia contains
backgrounds, fonts,
sounds and animations.
All you need to get started!

- ☐ Notice that the typeface and color are juust like the ones used on the previous page!



Make one more page

We want to make one more page to demonstrate different text wipes. Each line can have a different wipe.

- ☐ Press the F3 function key to create one more page with the same background.
- ☐ Type the following text:

Scala MultiMedia has
many text wipes

Diagonal
Push with ease-in
Random pattern
Horizontal wipe

Scala Multimedia has many text wipes

Diagonal
Push with ease-in
Random pattern
Horizontal wipe

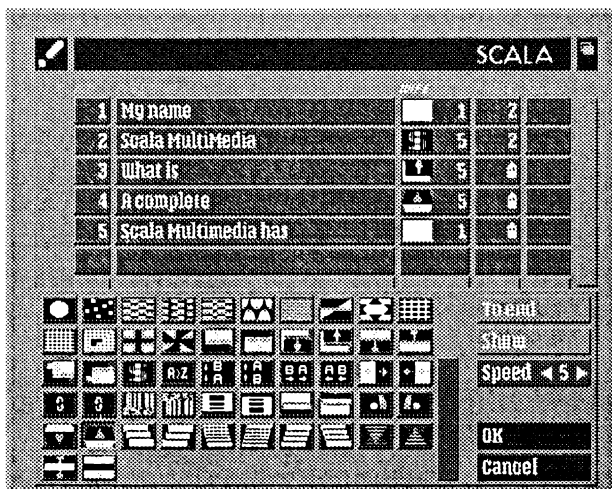
Text wipes

As we apply the appropriate text wipe to each line, you will see how you can apply wipes to many apges quickly. During the following steps, you can use the *Wipe speed* button to change the speed of the wipe, and use the *Show* button to check your results.

- ☐ Move the cursor to the "Diagonal" line, and click the *Wipe in* button in the *Edit* menu. The *Wipe in* menu appears. Select this wipe.
- ☐ While the *Wipe in* menu is still visible, move to "Push with ease-in", and select this wipe.
- ☐ Move to "Random pattern" and select this wipe.
- ☐ Select *OK* in the *Wipe in* menu. The last line already has a horizontal wipe!
- ☐ Select *Show* in the *Edit* menu to see how the wipes work. If you are not satisfied with any of the wipe speeds, move the cursor to this line and select the *Wipe in* button to edit the wipe.



TUTORIAL: ROUND TRIP



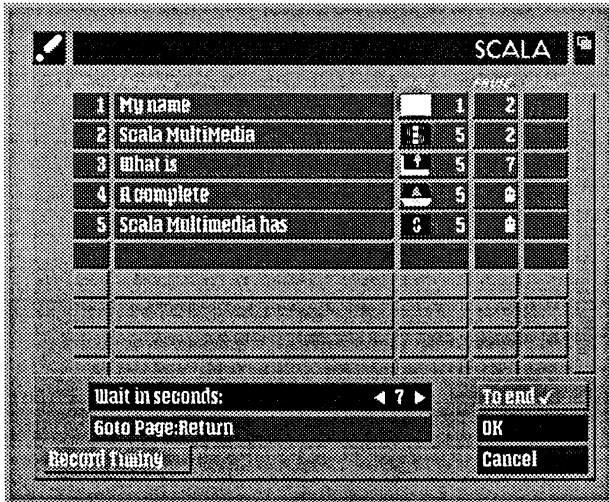
More Page wipes

We will now finish this page and select which wipes and pauses to use between the pages.

- ☐ Select OK to save the page, and select OK in the *Name page* menu.
- ☐ The *Main* menu appears.
- ☐ Select the *Wipe* button for the page called "What is". The *Page wipe* menu appears.
- ☐ Select the pop-up page wipe.
- ☐ While the *Page wipe* button is still displayed, select the *Page wipe* button for the page "A complete". The *Page wipe* menu is updated to show the settings for this page.
- ☐ Select the cube wipe.



TUTORIAL: ROUND TRIP



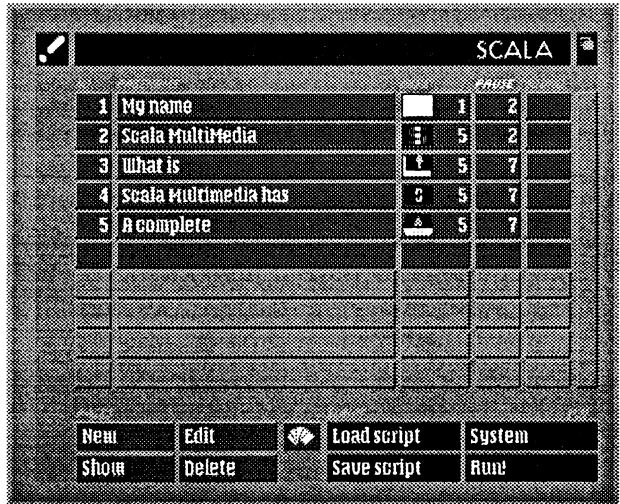
- ☐ Select the *Page wipe* button for "Scala MultiMedia has".
- ☐ Select the coin-flip wipe.
- ☐ Select OK in the *Page wipe* menu.



Setting the pauses

- ☐ Select the *Pause* button for the "What is" page. The *Pause* menu appears.
- ☐ Set the pause to 7 seconds.
- ☐ Turn on the *To end* option.
- ☐ Select OK.
- ☐ Select *Run* to view the script.

TUTORIAL: ROUND TRIP

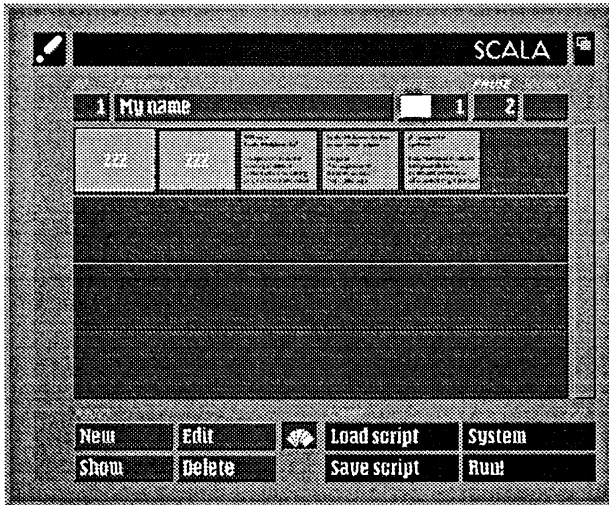


Moving a page

At any time, you can move pages up or down in the script. Now we are going to move the page containing the page wipes between the two pages above it.

- ☐ Select the page named "Scala MultMedia has", and hold down the left mouse button.
- ☐ Drag the page up one line, so that it is between the lines "What is" and "A complete".
- ☐ Release the mouse button.
- ☐ Select *Run* to view the script.

TUTORIAL: ROUND TRIP



Enter the *Shuffler*

There are two ways to view the script list in Scala Multimedia. Up to now, you have been using the *List view*, but now

- ☐ Select the shuffler symbol, in the
- ☐ Select (click once) one of the pictures in the *Shuffler*. Notice that the settings on the script line above the miniature pictures show information about the picture you select.
- ☐ Double-click a miniature picture. The selected picture will be displayed, together with any text wipes etc. that have been set for this picture. Hit the *Escape* button to return to the *Shuffler*.
- ☐ Move the picture "Scala MultiMedia" back to the end of the script. This is done by simply dragging the picture to its new position, just like in the *List view*.



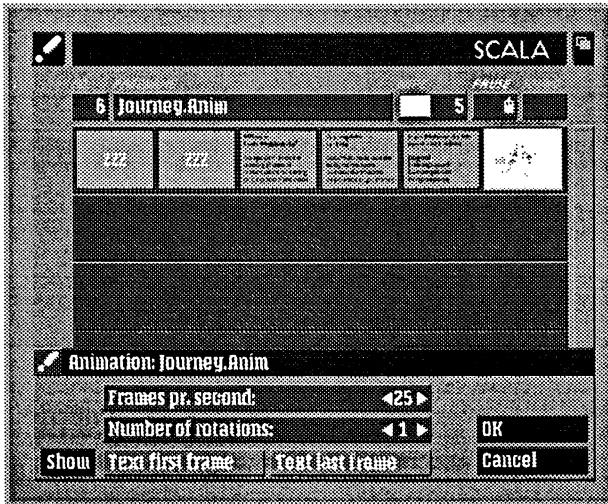
Loading a palette

The colors in the Scala Backgrounds are organized in a way that makes it easy to change them. In the "Palettes" drawer there are a number of pre-defined palettes with matching colors, made by the same designer that made the backgrounds. When you use these professionally designed palettes, you are guaranteed an attractive result.

Of course, you can use the *Color palette* to adjust each color yourself. See the reference section for more information about the *Color palette*.

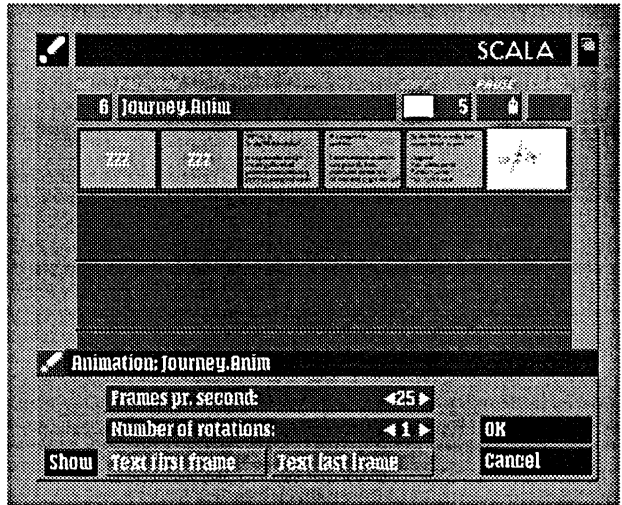
- ☐ While still in the *Shuffler*, select the picture with the name "A complete".
- ☐ Select *Edit*. The picture appears.
- ☐ Select *Palette* to bring up the *Color palette*.
- ☐ Select *Load* in the *Color palette*.

TUTORIAL: ROUND TRIP



- ☐ In the *File* menu, click the *Palettes* button.
- ☐ Select the palette called "YellowGold", and click *OK*. Notice that the colors in the picture and the color bar change.
- ☐ Back in the *Color palette*, click *OK* to go back to the *Edit* menu.
- ☐ Click *OK* to save the picture, and also click *OK* in the *Name page* menu. You are back in the *Main* menu.
- ☐ Run the script to see how it looks!

TUTORIAL: ROUND TRIP



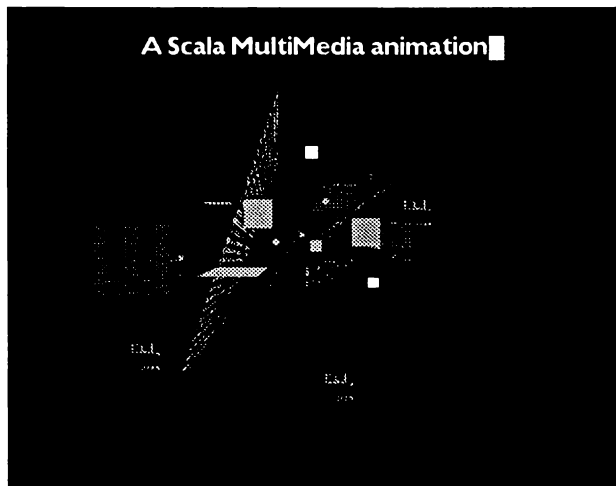
Adding an animation

Scala MultiMedia can play back animations. These animations can be made in a paint program, like Deluxe Paint IV. You can not create animations with Scala MultiMedia, but you can load and play any animation that follows the ANIM standard.

We are now going to load an animation located in the "Animations" drawer.

- ☐ Select *New* to create a new page. The *File* menu appears.
- ☐ Select the *Animations* button to the right.
- ☐ Select the animation called "Journey.anim".
- ☐ Select *OK* in the *File* menu. The *Animation* menu appears. This menu is used to set the playback speed and the number of times the animation is to be repeated (rotations).

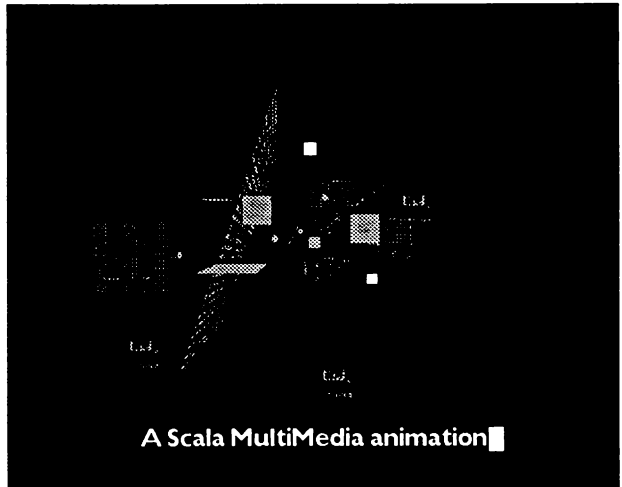
TUTORIAL: ROUND TRIP



- ☐ Set the number of frames per second to 25.
- ☐ Set the number of rotations to 8.
- ☐ Select *Show* to check that the animation runs correctly. You can use the *Escape* key to abort playback.
- ☐ Select *OK*.
- ☐ Set the *Pause* time for this animation to zero.
- ☐ Set the wipe for this animation to the one shown to the right.
- ☐ Run the script by selecting *Run!* in the *Main* menu.



TUTORIAL: A ROUND TRIP

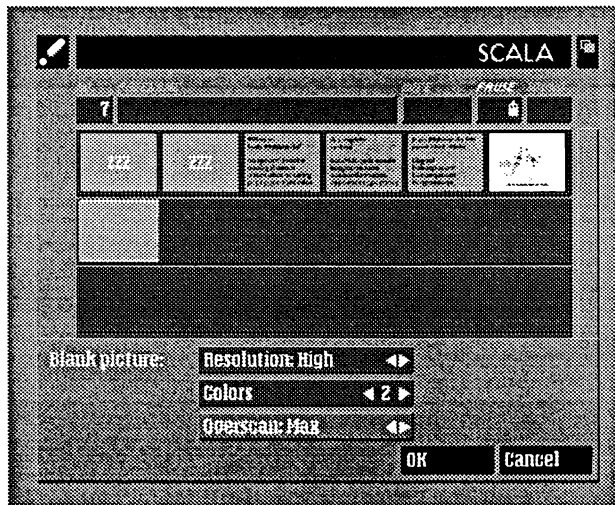


Adding text to the animation

In Scala MultiMedia, it is possible to add text to the first and the last page of an animation. Because of the way animation playback works, the text you put on the first frame of the animation can be destroyed as the animation is shown. The only safe areas are the ones that are not changed during playback. The last frame, however, can be titled with no such restrictions.

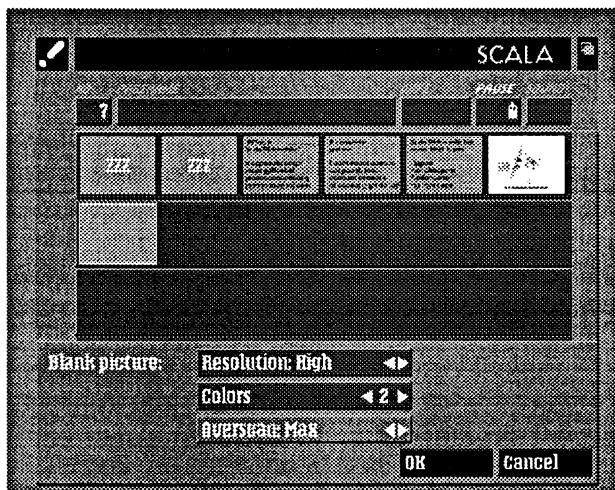
- ☐ Select the "Journey.anim" page in the *Main* menu
- ☐ Select *Edit*.
- ☐ Select *Text first frame*. The first picture in the animation and the *Edit* menu appears.
- ☐ Select the font *Gill 24*.
- ☐ Write the text "A Scala MultiMedia animation".
- ☐ Select center justification.

TUTORIAL: A ROUND TRIP



- ☐ Select *Move*, and move the text to the bottom of the screen.
- ☐ Select *OK* in the *Edit* menu.
- ☐ Select *OK* in the *Name page* menu, and you are back in the *Animation* menu.
- ☐ Select *OK* in the *Animation* menu, and the *Main* menu appears.
- ☐ Select *Run!* to see your script!

TUTORIAL: A ROUND TRIP



Making a text crawl

A special feature in Scala is the option to scroll long text lines horizontally across the screen. You can type in a long text, but before you start typing (or before the text reaches the edge of the screen), you need to turn on crawl in the *Text wipe* menu. Then the text will scroll to the left as you type.

- ☐ Select *New* in the *Main* menu.
- ☐ We are going to make a blank picture, so select *OK* in the *File* menu without selecting a file. The *Blank page* menu appears.
- ☐ In the *Blank picture* menu, select a Hi-res screen with two colors and maximum overscan. The reason for using overscan is that we want the text to move off the screen and not disappear behind a border.
- ☐ Click *OK* in the *Blank picture* menu.



- ☐ Select typeface FranklinC 114 in the *Font* menu.
- ☐ Turn on center justification.
- ☐ Type "IF YOU HAVE A" (*RETURN*) and "LONG MESSAGE," (*RETURN*).
- ☐ Select the crawl wipe with speed 10 in the *Text wipe* menu.
- ☐ Type "YOU CAN MAKE YOUR LETTERS FLY ACROSS THE SCREEN, JUST LIKE THIS" in one line, without pressing the *RETURN* key. The text will scroll left and off the screen as you type.
- ☐ Select OK to come back to the *Main* menu.
- ☐ Select the "T2" wipe with speed 10 for this page, and set the pause to 0.



TUTORIAL: A ROUND TRIP



The last page

We want the last page to look like the first one, but with a different text. To do this, we want to copy the first page, erase the text, and type in a new one.

- ☐ Enter the *List view* by clicking the *Shuffler* button.
- ☐ Click the page number of the page that has your name. The *Page control* menu appears.
- ☐ Click the *Copy* button.
- ☐ Set the *To after page* to 7.
- ☐ Click *OK*, and the page is copied to the end of the script. The name of the page will be your name, plus the suffix ".1".
- ☐ Double-click the name of the copy you just made. This is a shortcut for selecting the page, and then selecting *Edit*. The *Edit* menu appears.

TUTORIAL: A ROUND TRIP

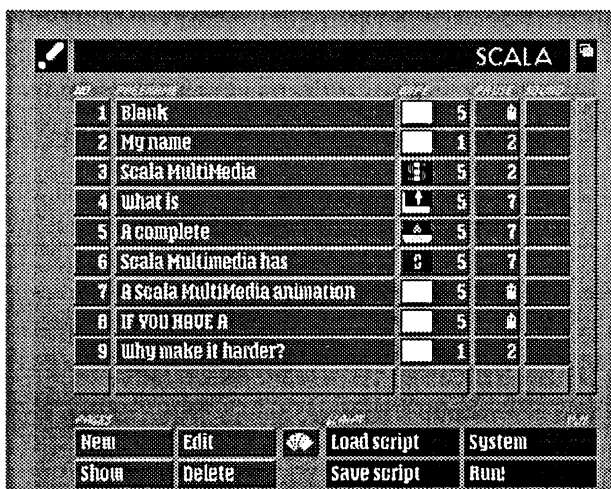


Why make it
harder?

- ☐ The cursor is at the start of the first line. Hold down the *SHIFT* key and press the *DEL* key to erase the rest of the line.
- ☐ Press the cursor *DOWN* key to move to the line below, and erase the text on this line too.
- ☐ Press the cursor *UP* key to move to the first line.
- ☐ Type "Why make it" and press *RETURN*.
- ☐ Type "harder?".
- ☐ Select *OK* in the *Edit* menu.
- ☐ When the *Name page* menu appears, change the name to "Why make it harder?", and click *OK*.
- ☐ Set the *Pause* to 6 and the wipe to the one shown to the right, with speed 10.
- ☐ Select *Run!* in the *Main* menu to view the script.



TUTORIAL: A ROUND TRIP



Adding some blank pages

Maybe you noticed that the first page of the script did not fade in, as we had selected. To make this happen, you need to add a blank page at the beginning of the script.

- ☐ Select *New* in the *Main* menu to create a new page.
- ☐ Since we are about to make a blank page, select *OK* in the *File* menu without selecting a file first.
- ☐ In the *Blank page* menu, select a hi-res page with two colors and no overscan, and select *OK*.
- ☐ When the *Edit* menu appears, select *OK* without typing anything first.
- ☐ The page will be named "Blank" in the *Name* page menu. Click *OK* to save the page and go back to the *Main* menu.
- ☐ Drag this page to the top of the script.

TUTORIAL: A ROUND TRIP



- ☐ To make a copy of the blank page, click the *Page number* button for this page. The *Page control* menu appears.
- ☐ Select *Copy* in the *Page control* menu.
- ☐ Use the *increase/decrease* arrows to set the "To after page" to the last page in the script.
- ☐ Select *OK*. The page is copied to the end of the script, and it gets the name "Blank.1".
- ☐ Set pause 0 for both the blank pages.
- ☐ Set the *Fade* wipe for page number two and page number 10.

TUTORIAL: A ROUND TRIP

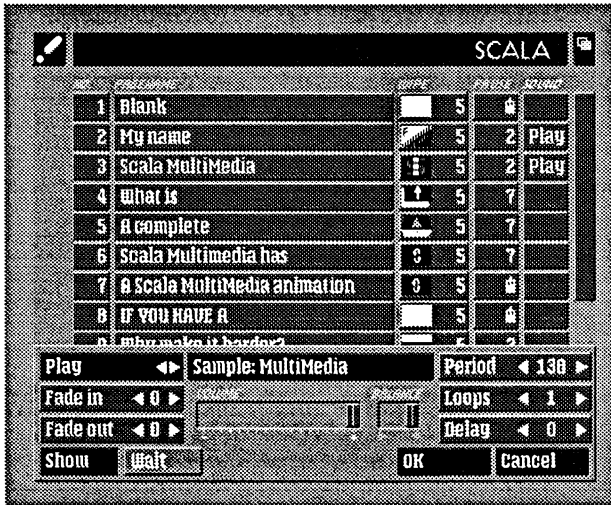


Making Scala Multimedia speak

Now it's time to add some sound! When the page that has your name is displayed, we want the Amiga to say "Scala", and when the next page is displayed, the Amiga should say "MultiMedia".

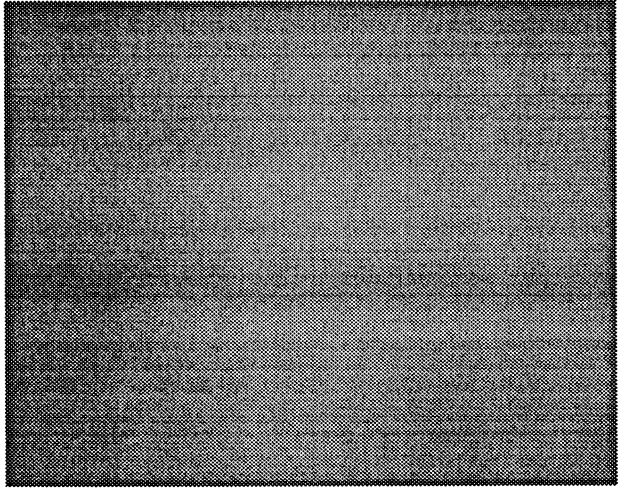
- ☐ Select the *Sound* button for page number two (the page with your name). The *Sound* menu appears.
- ☐ Select the *Load sound* button to see the *File* menu.
- ☐ Select the *Sounds* button to the right, and the names of the available sounds are displayed.
- ☐ Select the sound called "Scala", and select OK. Back in the *Sound* menu, you see some new buttons.
- ☐ Select *Show* to hear the Amiga say "Scala".
- ☐ To make the voice sound a bit deeper, decrease the *Pitch* value a bit. Use *Show* to test how it sounds.

TUTORIAL: A ROUND TRIP



- ☐ Move the *balance selector* to the left, to make the sound come from the left speaker.
- ☐ Select the *Sounds* button for the next line. The menu changes to the reflect the sound setting for this line.
- ☐ Select *Load Sound*. The file menu appears.
- ☐ Select the sound called "MultiMedia", and click OK.
- ☐ To make also this the voice sound a bit deeper, decrease the *Pitch* value a bit.
- ☐ Move the channel selector to the right, to make the sound come from the right speaker.
- ☐ Select OK to return to the *Main* menu.
- ☐ Select *Run* to see (and hear!) the script!

TUTORIAL: A ROUND TRIP

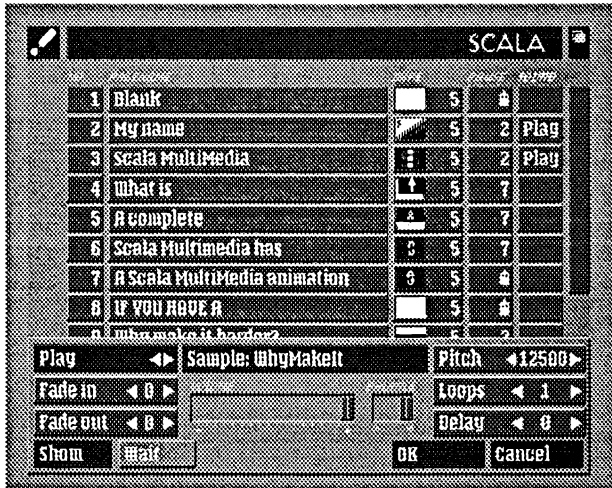


The sound of music

To play some music, we are now going to load a soundtracker module that will play while the script is shown. The music will start from page number four.

- ☐ Press the *Sound* button on the line named "What is". The sound menu appears.
- ☐ Select *Load* sound.
- ☐ In the *File* menu, select the *Music* button on the right side to see the contents of the *Music* drawer.
- ☐ Select the song called "mod.Techno", and select *OK*.
- ☐ Select *OK* in the *Sound* menu.
- ☐ Select *Run* in the *Main* menu to run the script. The music should now start playing when the page "What is Scala Multimedia?" pops up!

TUTORIAL: A ROUND TRIP

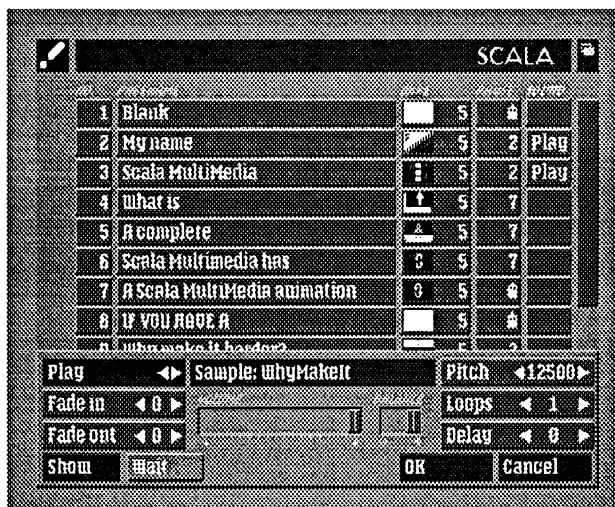


Why make it harder?

At the end of the script, when the page "Why make it harder?" is displayed, we want Scala MultiMedia to say "Why make it harder?". This question is also located in the "Sounds" drawer. We also want the sound to fade out at the end.

- ☐ Select the *Sound* button for the page "If you have a".
- ☐ In the *Sound* menu, use the arrows on the top left button (the command selector) to select the command *Stop*.
- ☐ Set the *Fade* value to 8 seconds.
- ☐ Select the *Sound* button for the page "Why make it harder?".
- ☐ Select *Load Sound*.

TUTORIAL: A ROUND TRIP



- ☐ In the *File* menu, select the sound called "WhyMakeIt" that is located in the "Sounds" drawer, and select *OK*.
- ☐ Select *OK* in the *Sound* menu.
- ☐ Select *Run* to run the script.

This is the end of tutorial one!

TUTORIAL: A ROUND TRIP

Tutorial two: The number quiz

In this tutorial you will learn how to use buttons and variables. These features, combined with the flexibility of the EX system, open up a world of exciting possibilities.

We are going to make a small game. The object of the game is to guess a number. When you run the finished script, you see a row of numbers from one to nine. When you select a number, you see a "Too high" or "Too low" message, and the counter that counts how many tries you have had is increased by one. If you answer correctly, you are congratulated!

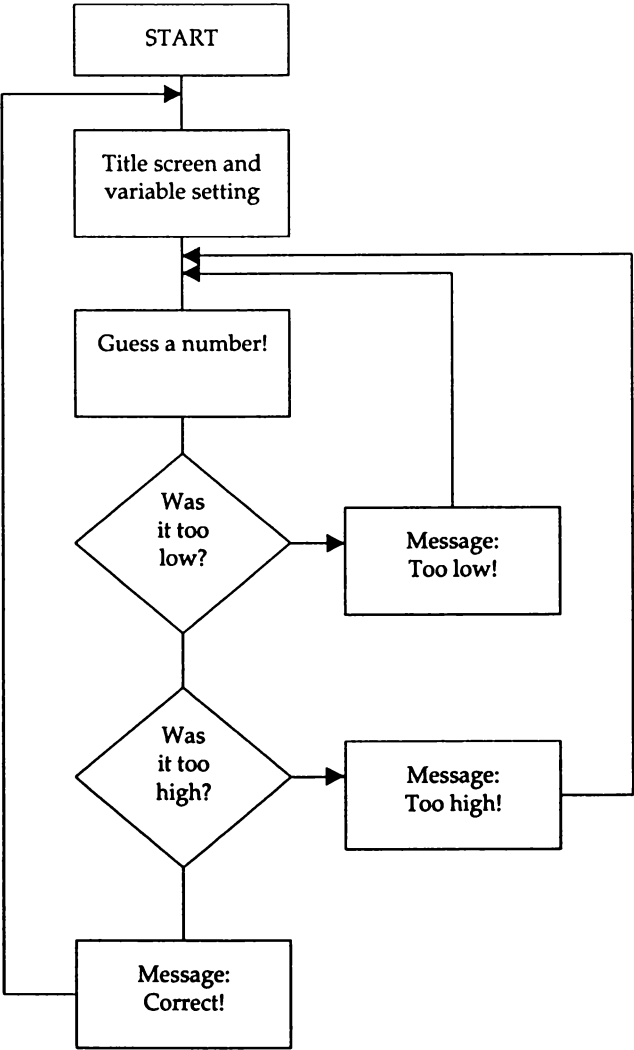
On the next page you will find a flow chart showing how the script is supposed to work. Since scripts with buttons and variables often require some planning, we suggest that you make a flow chart before starting. This will make it easier to get started on a complex job.

The following variables are used in the flow chart, and will also be used in the script.

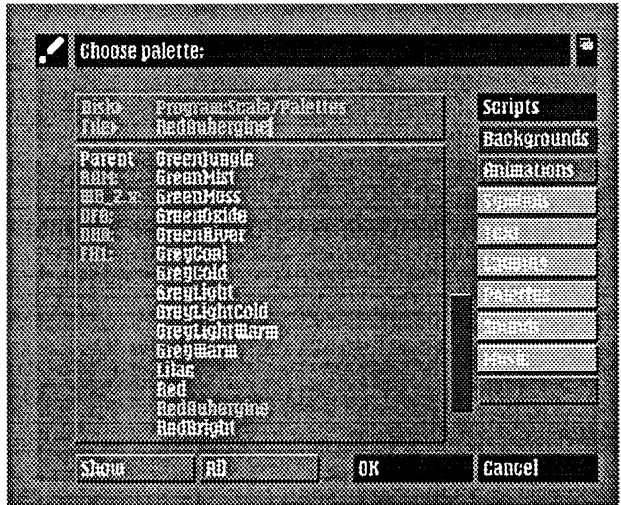
n	A random number from one to nine
g	The number that the user has guessed
t	The number of guesses

Variables can have long names, but we have used one-letter variable names in this example.

TUTORIAL: THE NUMBER QUIZ



TUTORIAL: THE NUMBER QUIZ

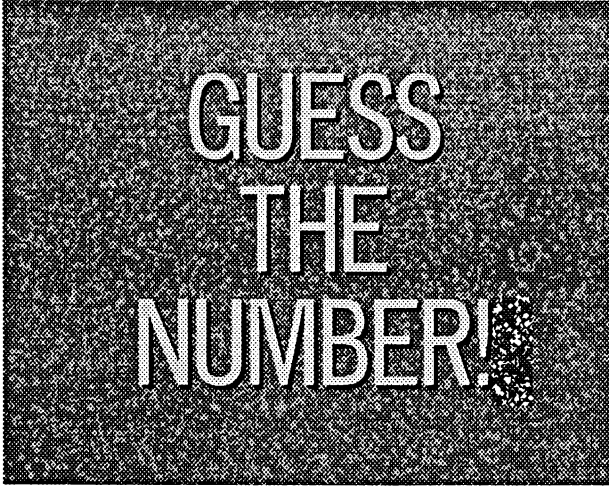


Making the intro page

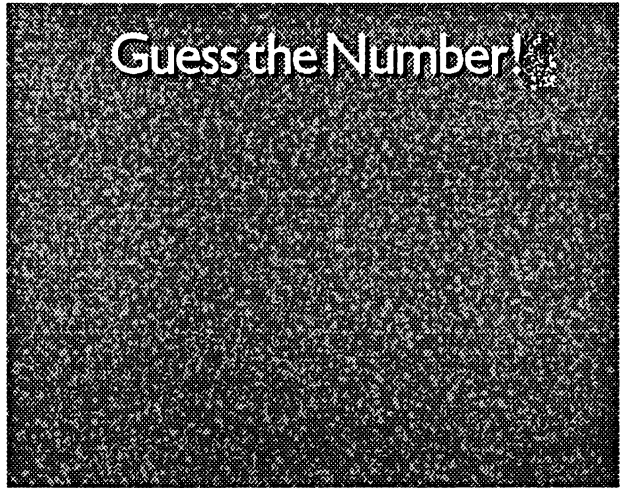
This page is a "title page" for the game. It could contain instructions on how to play the game, credits etc., but we will just have a title on it. Later in this tutorial, this page will also be used to set some variables.

- ☐ Start Scala MultiMedia.
- ☐ Select *New* in the *Main* menu.
- ☐ Select the background Texture010 in the *File* menu, and select *OK*. The background picture appears.
- ☐ Select *Palette*.
- ☐ Load the palette "RedAubergine" that is located in the "Palettes" drawer.
- ☐ Select typeface NewsGothic 114 in the *Font* menu.

TUTORIAL: THE NUMBER QUIZ



- ☐ Select front color orange.
- ☐ Turn on center and shadow.
- ☐ Type
GUESS
THE
NUMBER!
- ☐ Select *Move*, and move the text a bit down.
- ☐ Select *OK*, and save the page. Give it the name "Intro".

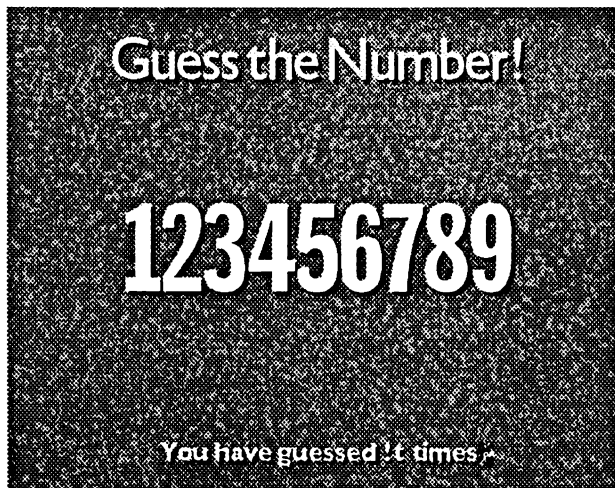


The game page

This page will contain a row of numbers, and buttons that the user can select when he makes a guess.

- ☐ Select *New*.
- ☐ Select the background Texture010 in the *File* menu.
- ☐ Select *Palette*.
- ☐ Load the palette "GreyLightWarm".
- ☐ Select typeface GillIN 58.
- ☐ Turn on center and shadow.
- ☐ Select front color orange.

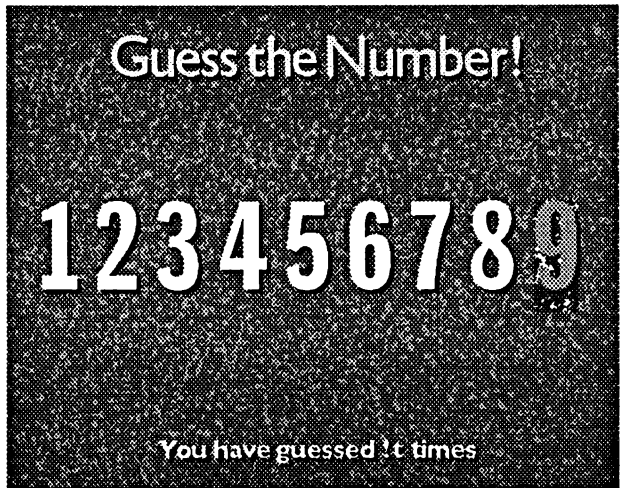
TUTORIAL: THE NUMBER QUIZ



- ☐ Type "GUESS THE NUMBER!" across the top of the page.
- ☐ Move the cursor to the middle of the screen.
- ☐ Select typeface FranklinC 114.
- ☐ Select front color red.
- ☐ Type "123456789".
- ☐ Move the cursor to the bottom of the screen.
- ☐ Set the *Front* color to light yellow.
- ☐ Select typeface Gill 24.
- ☐ Type "You have guessed !g times"

The expression "`!g`" is a variable. If you put the exclamation mark in front of a variable name, it will be printed on the screen when you run the script.

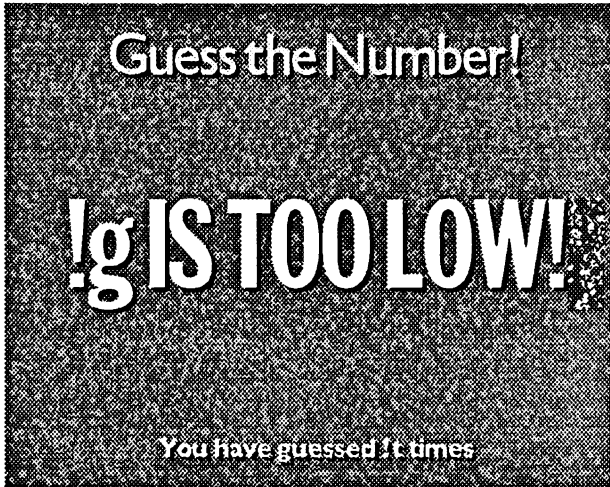
TUTORIAL: THE NUMBER QUIZ



- ☐ Move the cursor up to the "123456789" line.
- ☐ Hold down the CTRL key and press the right-arrow cursor key a few times, to increase the character spacing so that the line is almost as wide as the screen.

Later we will create a button for each number, but first we are going to make all the pages in the script. These are the three pages that contain the "too high!", "too low!" and "correct!" messages.

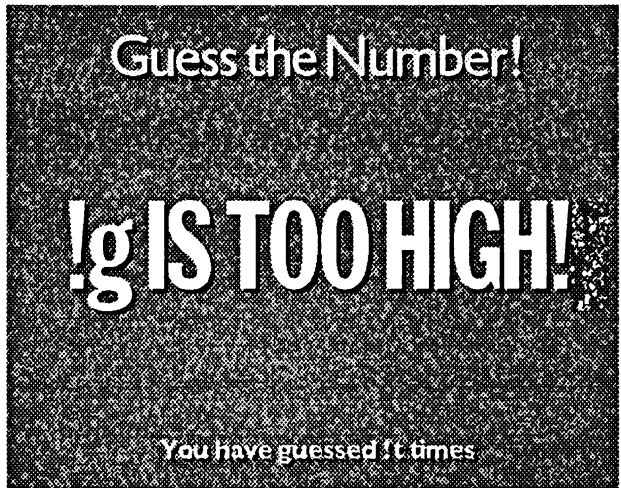
TUTORIAL: THE NUMBER QUIZ



The "Too low!" page

This page will be very similar to the "Guess the number" page that you are now editing. Therefore, we use the *F4* function key to create a copy of the page with the same background and text as this one.

- ☐ Press *F4* to create a new page with the same text. You are now working on a copy of the original page.
- ☐ Move the cursor to the "123456789" line.
- ☐ Enter the *Layout* menu, and set the character spacing to 0.
- ☐ Select *OK* in the *Layout* menu.
- ☐ Press *Amiga-X* to delete the line.
- ☐ Type "!g IS TOO LOW!"

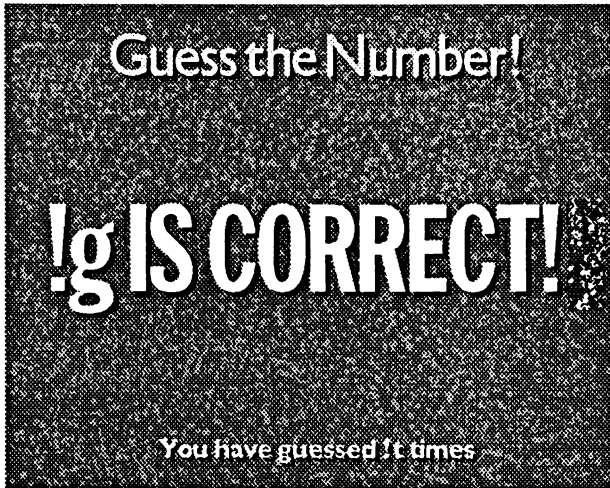


The "Too high!" page

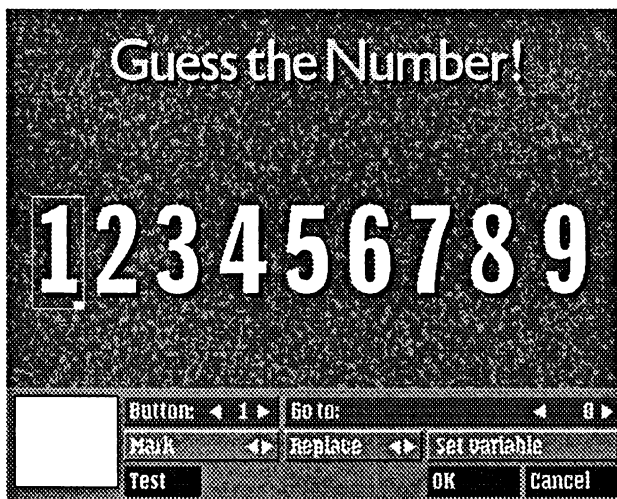
This page will be very similar to the "Too low!". We will use the *F4* function key once more to create a copy of the page with the same background and text as this one.

- ☐ Press *F4* to create a new page with the same text.
- ☐ Move to the "!g IS TOO LOW!" line.
- ☐ Move the cursor between "TOO" and "LOW".
- ☐ Press *SHIFT+DEL* to delete the rest of the line.
- ☐ Type "HIGH", so that the whole line reads "!g IS TOO HIGH!"

TUTORIAL: THE NUMBER QUIZ



- ☐ Press F4 again to create a new page with the same text.
- ☐ Move to the "!G IS TOO HIGH!" line.
- ☐ Move the cursor between "IS" and "TOO".
- ☐ Press SHIFT+DEL to delete the rest of the line.
- ☐ Type "CORRECT!", so that the whole line reads "!G IS CORRECT!"
- ☐ Select OK, and save the page. Give it the name "CORRECT!"

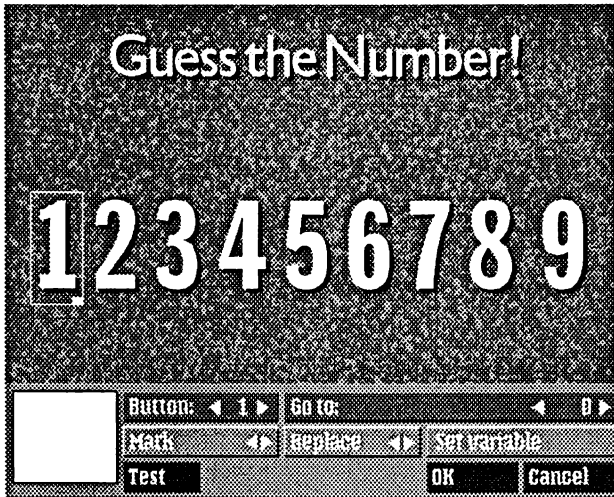


Making some buttons

Now we will make Scala buttons! All the buttons in the script will be located on the "Guess the number!" page, and there will be one button for each of the numbers you have made.

- ☐ In the *Main* menu, select the page "Guess the number!"
- ☐ Select *Edit*.
- ☐ When the *Edit* menu appears, click the *Button* button. You now see the *Button* menu.
- ☐ To make a button that covers the "1", move the pointer above and left of the number, press the left mouse button, and drag down and right until the whole "1" is inside the rubberband.

TUTORIAL: THE NUMBER QUIZ

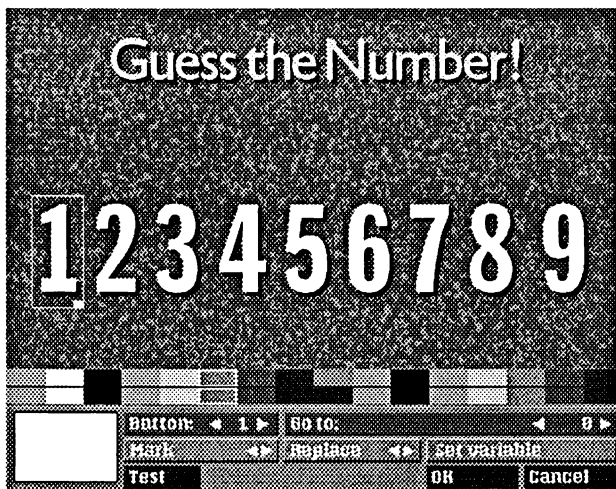


You have now made your first button! A button can be moved by pointing inside it and dragging it to its new position. To change its size, drag the "handle" in the lower right corner to its new position. To remove a button, select it, and then press the DEL key.

All buttons on one page are numbered from one upwards. The number and settings for the selected button are displayed in the *Button* menu. To select a button, click inside it with the left mouse button, or use the left/right arrows on the *Number* button to cycle through the list.

Button appearance when marked

When the user moves the pointer on to the button, something has to change to show that the button is being pointed at (marked). We will make the color turn yellow when marked.



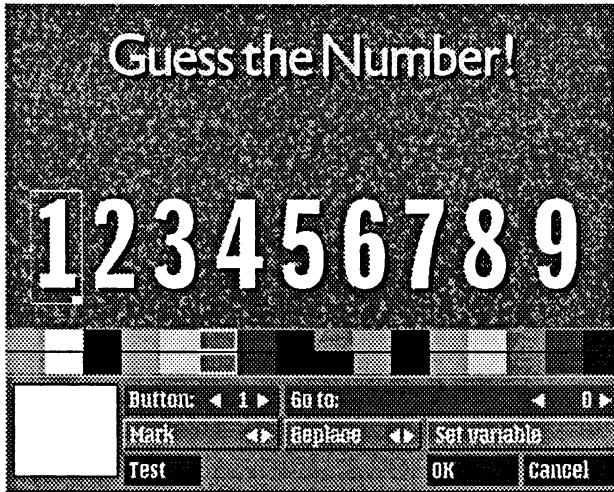
- ☐ Set *Mark/Select* to *Mark*.
- ☐ Set the highlighting to *Replace*.

Replace is a highlighting method that can replace colors inside the button square with other colors. You can select which colors to change, and what colors they shall change to.

A double color bar appears above the *Button* menu. This color bar is used to select which colors that should be replaced when the button is highlighted. The upper row contains all the colors in the palette, while the bottom row contains the colors that the ones above are going to change to.

To change colors in the bottom row, first select the wanted color in the upper row, and then select the square in the bottom row you want to change. The color you selected in the upper row will be copied to the square.

TUTORIAL: THE NUMBER QUIZ

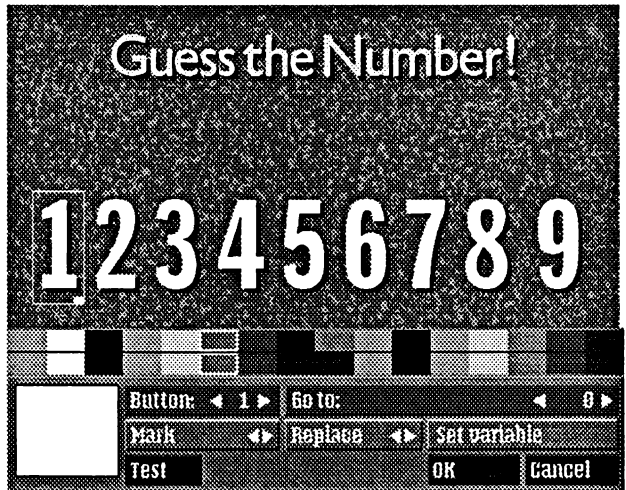


When the pointer moves across the button you made, we want the color of the number to become yellow. This means that we have to copy the yellow color in the upper row, to the square below the letter color.

- ☐ Select the warm yellow color in the upper row.
- ☐ Select the square below the color of the letters.

The yellow color is copied to this square, which means that the letter color will change into warm yellow when the pointer touches the button you have made. Technically speaking, all "number-colored" pixels inside the button square will be replaced by yellow ones.

TUTORIAL: THE NUMBER QUIZ

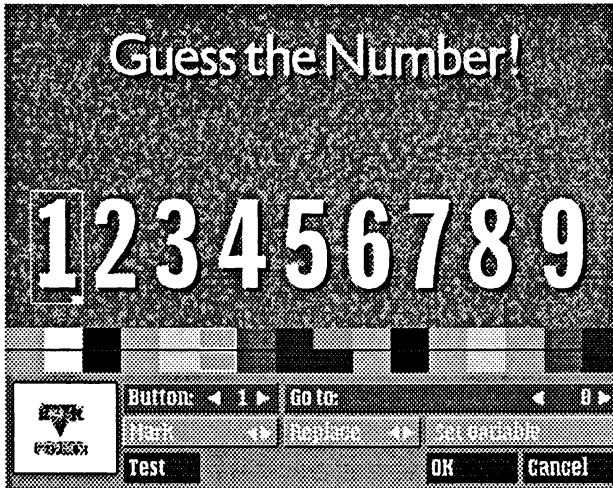


Button appearance when selected

When the user clicks the mouse button to choose a number, the button must change again to confirm the users' action (selected). We will make the number turn white when selected.

- ☐ Set the *Mark/Select* button to *Select*.
- ☐ Set the highlighting to *Replace*.
- ☐ Use the same procedure as above to make the letter color change to white. This means that the square below the red color must be white.
- ☐ To see if the button works as intended, select *Show*. Move the pointer to the "1". When the pointer touches the square (the square itself is invisible), the number changes color to yellow. When you click the mouse button, the color changes to white! Press the *ESCAPE* key to get back to the *Button* menu.

TUTORIAL: THE NUMBER QUIZ

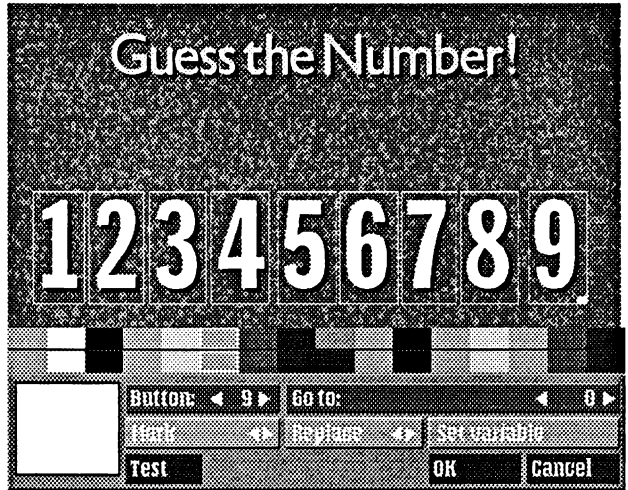


Variables

As mentioned earlier, we use variables in this script. When the user clicks a button (a number), Scala MultiMedia will need to remember which number was selected, in order to check if it was correct. In the *Button* menu you have a button called *Set variable*. This option can be used to change the value of one or more variables if the user has selected the specified button. If you select the *Set variable* button, the *Variable* menu appears. You can then select a variable (or create a new one), and decide its value. These operations will only be performed if the user has selected *this* button.

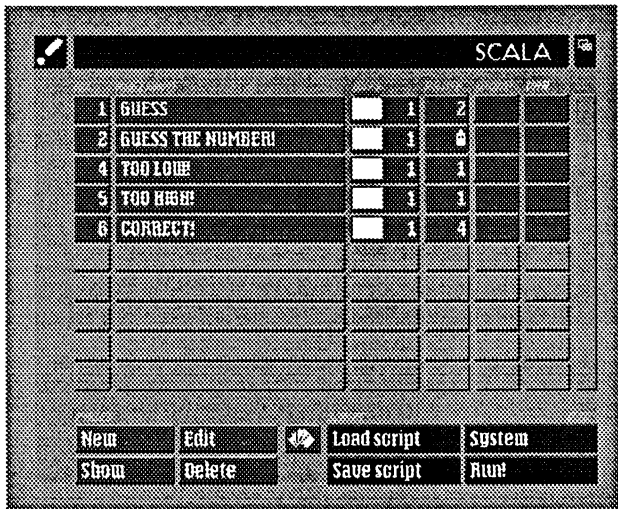
- ☐ Select the *Set variable* button.
- ☐ In the *Variable* field, type the variable name "g", and press RETURN. "g" is short for "guess", and this variable will contain the number that the user guessed.
- ☐ In the *Set* field, type "1". Since this button indicates that the user selected number 1, we set "g" to this value.

TUTORIAL: THE NUMBER QUIZ



- ☐ Select *OK* in the *Variables* menu to get back to the *Button* menu.
- ☐ Now, create a button that surrounds "2", and repeat the steps above for this button. The *Mark/Select* settings are copied from the last button you made, so you don't have to repeat *all* the steps! The replace colors will be the same as on the first button you made. In the *Variables* menu, set "g" to "2" for this button.
- ☐ Do the same for the rest of the numbers up to 9, and remember to use the *Variables* menu to set the variable "g" to the number that the button covers.
- ☐ Select *Ok*, and go back to the *Main* menu.

TUTORIAL: THE NUMBER QUIZ

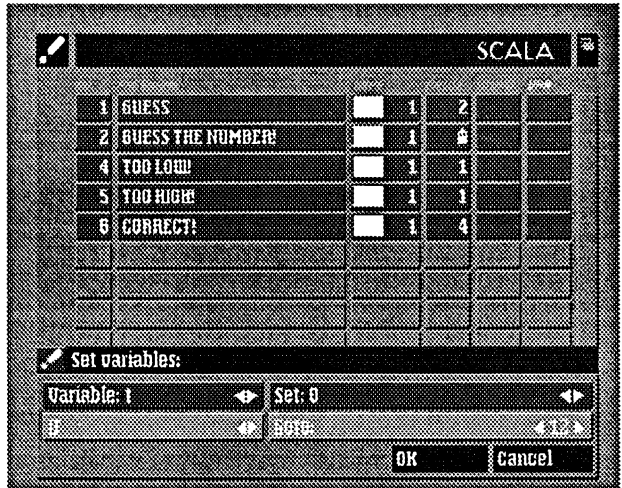


Variables again

Now we have an intro picture, a question picture where the variable *g* is set according to which buttons are pressed, and three "result" pages. What we need now is to "calculate" which page to jump to, based on which number the user has chosen.

In the *Main* menu, there is a column called *Variables*. If these buttons are not visible, you may have to change the width of some of the other columns in order to make room for the *variable* buttons. To make the *Pagename* column a bit narrower, move the pointer to the opening to the right of the column, and press the left mouse button. A vertical line appears, and you can now drag this line to the left in order to make the column less wide. If you have a lot of other EX modules in your system, you may have to rearrange the order of the columns to see the *Variables* column. This is done in the *System* menu - see the reference section for more information.

TUTORIAL: THE NUMBER QUIZ



Initializing

In the beginning of the script we need to set some of the variables to zero, and also select a random number between one and nine, which we will put in variable "n".

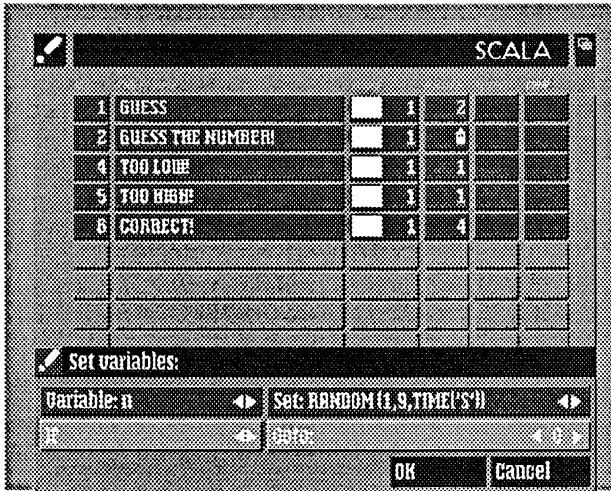
We put these instructions in the *Variables* column of the first page in the script.

- ☐ Select the *Variables* button on the blank line. The *Variables* menu appears.
- ☐ In the *Variable* field, write the variable name "t".

This is the variable that contains the number of tries the user has had. Since these instructions are performed *before* the user starts to guess, the number of tries must be set to zero.

- ☐ In the *Set* field, type "0". This means that the variable "t" has a value of zero.

TUTORIAL: THE NUMBER QUIZ



- ❑ To enter another variable, use the left/right arrows on the *Variable* button to get a blank field. Then type in the variable name "n".

The variable "n" will contain the secret number. This number must be chosen by chance each time the game is started we use the RANDOM function.

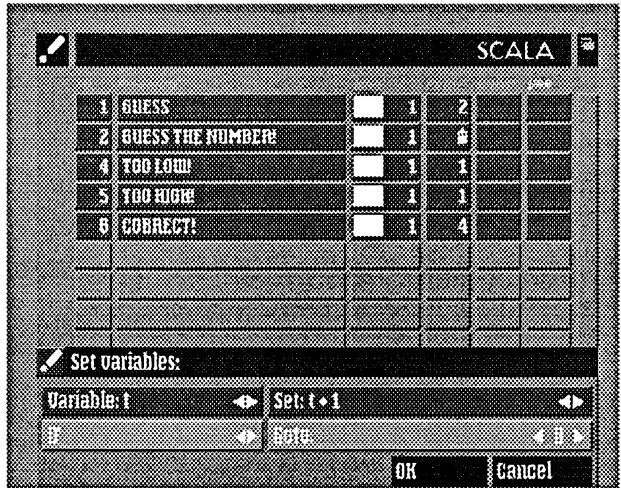
- ❑ To get a random number from one to nine, type the following in the *Set* field:

RANDOM (1,9,TIME('S'))

- ❑ Select OK to get back to the *Main* menu.

Each time this page is displayed, Scala MultiMedia will set the variable t to zero, and set n to a random number between one and nine.

TUTORIAL: THE NUMBER QUIZ



Checking the guess

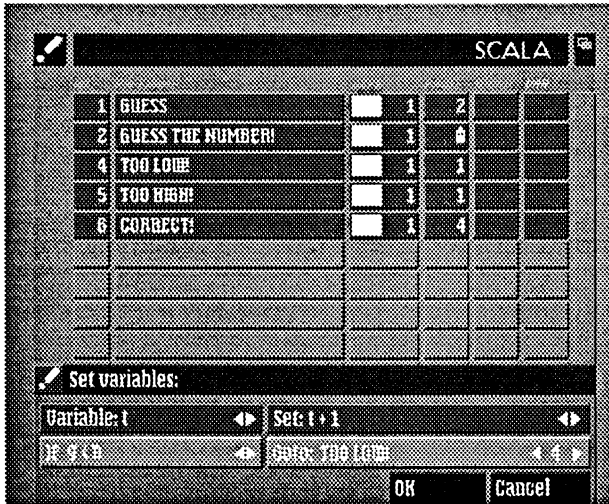
The input must be tested right after the user has selected a button. Therefore we put these instructions right after the "GUESS THE NUMBER!" page. But we don't want to display a picture or an animation while the test is done - the *Variables* column is the only one that will contain something on that line.

- ☐ To insert a blank line after the "GUESS THE NUMBER!" page, move the blank line at the bottom of the script to line three. Line three now contains a blank line.
- ☐ Select the *Variables* button on the blank line. The *Variables* menu appears.

When the user selects a number, it means that he has used one more try. Therefore, the counter variable *t* must be increased by one each time this page occurs.

- ☐ Find the variable "t" in the *Variable* button by using the left/right arrows on the button.

TUTORIAL: THE NUMBER QUIZ

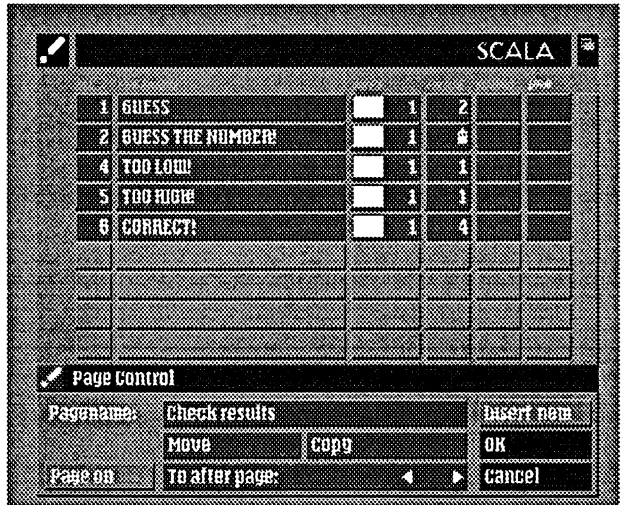


- ☐ In the *Set* field, type "t + 1"

Now we have to test the "g" variable. This is the number that the user guesses, and "n" is the correct number. If "g" is larger than "n", then the "too large!" message must be displayed.

- ☐ Select the *IF* button in the lower left corner. The cursor appears after *IF*. Type "g > n"
- ☐ Then select which page to jump to, using the left/right arrows on the *Goto* button to the right of the *IF* button. Select the page "TOO HIGH!". You have now decided that if "g" is larger than "n", Scala MultiMedia will jump to the page "TOO HIGH!" and continue from there.
- ☐ Use the left/right arrows on the *IF* button to select an open IF-test, and type "g < n"
- ☐ Select to jump to page "TOO LOW!"

TUTORIAL: THE NUMBER QUIZ



Was it correct?

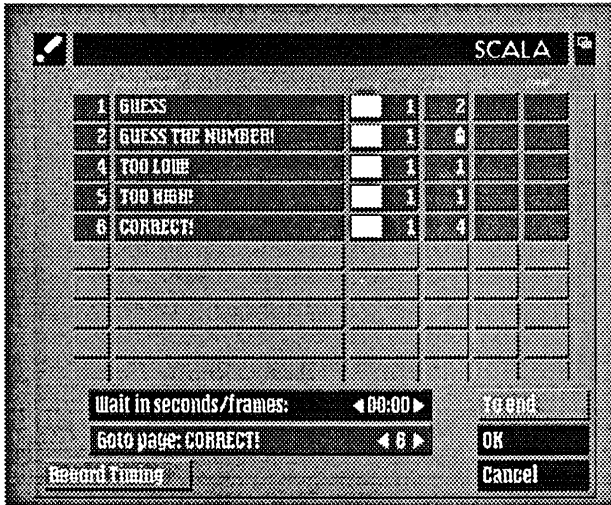
Now we could have added another test to check if "g" was equal "n", but those of you that are familiar with programming, will see that if Scala MultiMedia does *not* jump to either of the two pages, then the numbers must be equal. Therefore, we don't have to add a test for this, but simply use the *GOTO* statement in the *Pause* column.

- ☐ Select OK in the *Variables* menu.

The blank line you just made, has got the name "Unnamed.1". It is smart to have page names that tell what is happening. If not, large scrips can become difficult to edit. To rename this page to "Check results", do the following

- ☐ Select the *Pagenumber* button for page 3. The *Page control* menu appears.
- ☐ Erase the contents of the *Pagename* field, and type "Check results".
- ☐ Select OK to get back to the *Main* menu.

TUTORIAL: THE NUMBER QUIZ

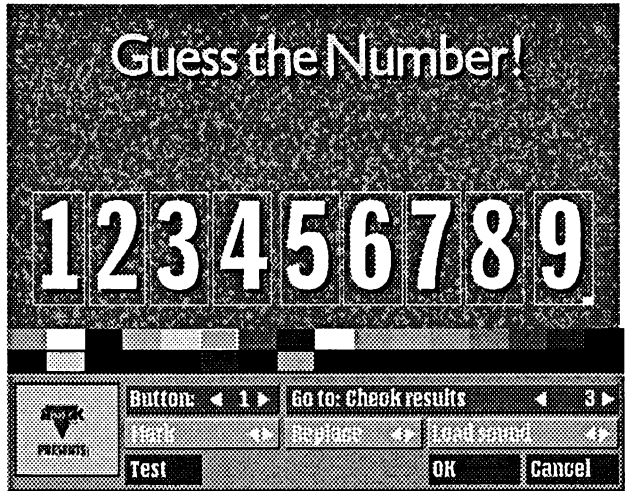


To handle the case where neither of the two *IF*-tests on line three were true, we have to go to page number six, which displays a "correct" message. This is done with the *Pause* button for page three.

- ☐ Select the *Pause* button for page number three.
- ☐ Set the *Goto page* to number six, "Correct!"

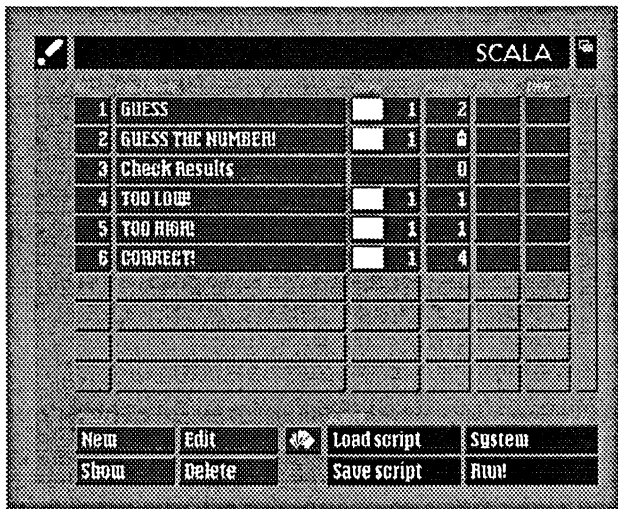
When the user has selected one of the nine buttons on page two, the variable "g" is set to the number of the button. But when one of the buttons has been pressed, we have to make Scala MultiMedia jump to the "Check results" page.

TUTORIAL: THE NUMBER QUIZ



- ☐ Select page two, and then select *Edit*. The "Guess the Number" page and the *Edit* menu appears.
- ☐ Select *Buttons*.
- ☐ Select button number one by clicking inside it, or using the left/right arrows on the *Button* button.
- ☐ Set the *Goto* page to page number three: "Check results"
- ☐ Do the same for all the other buttons on this page.
- ☐ Select *OK* and save the page.

TUTORIAL: THE NUMBER QUIZ



Now we have finished the basic structure

Page one: Intro

Displays a welcome screen and defines the variable *t* (number of tries) and sets *n* to a random number between 0 and 10.

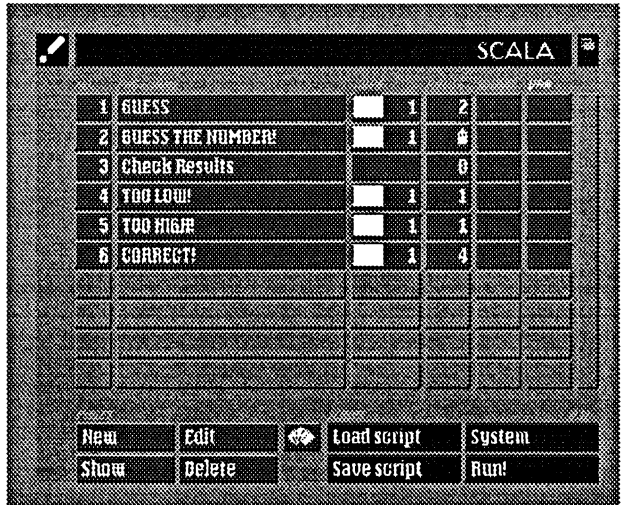
Page two: Guess the number!

Accepts a selection from the user. The variable "g" is set to a number between one and nine, according to the user's choice. Scala MultiMedia then jumps to page three.

Page three: Check results

The number of tries is increased by one. Then the number "g" is tested: if it is smaller than "n", Scala MultiMedia jumps to page four which displays the message "Too low". If "g" is larger than "n", then page five is displayed. If neither of these tests were true, then the numbers must be equal, so Scala MultiMedia jumps to page six.

TUTORIAL: THE NUMBER QUIZ



Page four: Too low!

This page displays the message "<number> is too low!"

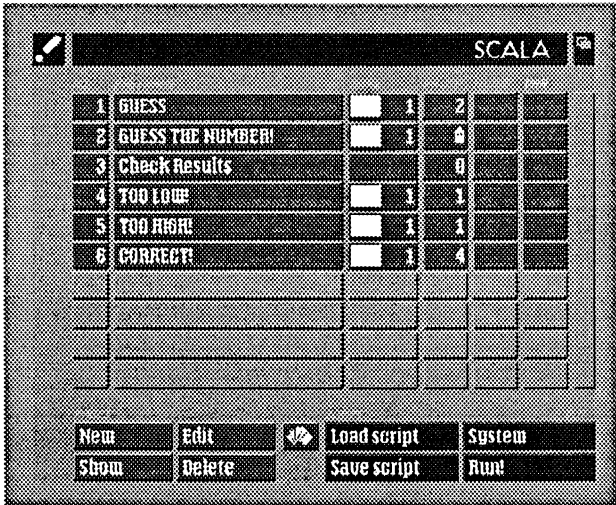
Page five: Too high!

This page displays the message "<number> is too high!"

Page six: Correct!

This page displays the message "<number> is correct!"

TUTORIAL: THE NUMBER QUIZ



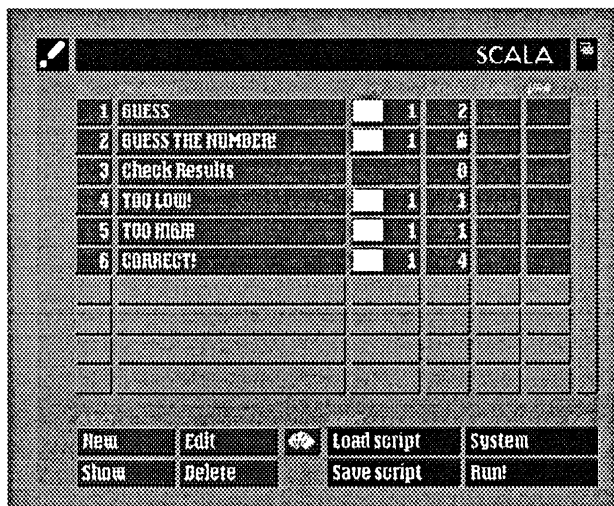
To clean up the last bits here, we need to set a *Goto* page for the three last pages. If the number is not correct, the user can guess once more. This means that the *Goto* for page four and five must be set to page number two: "Guess the number".

- ☐ Set the *Goto* number to two for page four and five in the *Pause* menu. Also set the pause for these two pages to two seconds.

If it was right, we need to find a new secret number and to reset the counter "t" to zero. This is done on page one in the script, and therefore we do a *goto* to this page.

- ☐ Set the *goto* number for page six to one (the "intro" page). Set the pause for this page to two seconds.

TUTORIAL: THE NUMBER QUIZ



Finished?

If everything has been done correctly, the quiz should now work according to the flow chart. If not, check to see if you have forgotten any variables, buttons or goto's. Interactive scripts may be a bit difficult to test before they are nearly completed, but proper planning will help you a lot (flowchart!).

To improve the script further, here are some ideas:

- Use more interesting wipes between the pages.
- Add sounds!
- Make an "Exit" button on page two
- Use other backgrounds etc.

If you experiment with this script, and also take a closer look on the "ScalaQuiz" demo script, you will soon know all the possibilities of buttons and variables. You will find that using Scala MultiMedia to create interactive applicatins is easier than any other way - so why make it harder?



Putting it all together

In this chapter we give you a series of practical tips and information about how to get the most out of Scala MultiMedia.

PUTTING IT ALL TOGETHER

What is multimedia?

In brief, it means to mix together different kinds of "output" to create something that gives a stronger impression than just a single picture or sound; a kind of "total" experience where more than one sense is stimulated.

For those of you who already have put together a show consisting of video, sound, user input etc., there is another definition that you will probably recognize: "Multimedia is a lot of cables".

Although Scala MultiMedia is a complete software package that contains all you need to create a multimedia presentation, you can also add your own elements. In this section of the manual we will explain a bit about how to make pictures, sounds and animations. We hope to give you some ideas and inspiration for your further work.

The Amiga is a truly Amazing computer. For years, it has been able to do all the things other manufacturers now are now beginning to add to their computers. While it just recently has become possible to run several programs at once on a well-equipped IBM-compatible PC, the Amiga has been capable of true multitasking since the very first version of the operating system back in 1985. The independent sound and graphics chips do their jobs without slowing the system down.

This makes the Amiga perfect for multimedia applications.

The IFF Standard

Unlike other computers, the Amiga has very standardized file formats for graphics and sound. This makes it easy to exchange data between different software packages. One word you certainly will see more than once is IFF. This is short for "Interchangeable File Format", and is most often used about pictures and animations. But there are IFF standards for sounds, text and other kinds of information, too.

The IFF standard goes all the way back to the early years of the Amiga, and it was developed by Commodore-Amiga and Electronic Arts (the creators of Deluxe Paint). Since then, it has been continuously developed to adopt technical improvements.

This is why you can grab a video image in one program, modify it in another, and display it in yet another program (like Scala MultiMedia). Or generate three-dimensional animations in Scenery animator, modify them in Deluxe Paint and show them in Scala MultiMedia.

The IFF standard has been expanded to handle 24-bit images, but you will need extra hardware to display them.

PUTTING IT ALL TOGETHER

Pictures

When you select *New* from the *Main* menu, you are taken directly to the *File* menu. Then you can load an IFF picture or animation.

Resolutions

The Amiga graphics hardware is currently able to display pictures with a maximum horizontal resolution of 640 pixels and a vertical resolution of 512 pixels (400 on an NTSC computer). In this resolution the maximum number of colors is 16. If the horizontal resolution is 320 pixels, the maximum number of colors is 32.

HAM and Extra Halfbrite

However, the Amiga has a couple of "special" resolutions that give you more colors. In *HAM* mode (Hold And Modify), you can have pictures with up to 4096 colors. This mode is ideal for color photographs and digitized video. But it has one disadvantage: because the color of a pixel in a HAM picture depends on the color of the pixel to the left of it, texting such pictures will seldom turn out well. But for showing pictures and digitized video sequences, HAM is rather good.

PUTTING IT ALL TOGETHER: PICTURES

Overscan

"Normal" computers have a border around the edge of the screen which not can contain text or graphics, but this is unacceptable for video applications. The Amiga graphics hardware is capable of showing pictures that are larger than the visible area of the screen. this feature is called *overscan*.

When you select *New* from the *Main* menu, the *File* menu will appear, asking you to select a background picture or animation. When you load a picture, the screen will always be sized to the picture size. This means that when you load a standard 640x512 image, there will be no overscan. When you install Scala MultiMedia, you have the choice to convert all the background pictures to overscan.

If you make your own pictures in Deluxe Paint, you must turn on overscan before you start painting. If you have already made a non-overscan picture, use Deluxe Paint IV to stretch the picture. Turn on overscan, and Deluxe Paint IV will ask if you want the picture stretched to fit the new page size. If you answer "yes", the picture will be resized. Remember, however, that any scaling of a picture may result in loss of quality.

If you select *OK* in the *File* menu without selecting a file first, you are asked to enter the dimensions of a new blank page. There is also an *Overscan* button on this menu. Unless you need to have text off the edge of the screen, overscan can be turned off. But for video titling, it will probably be an advantage to use overscan. The overscan choices are *Standard* and *Max*.

Interlace flicker

The Amiga graphics chips are constructed to be compatible with video signals, and they have most of the honor for the Amiga's success in the video market. But video compatibility also causes a disadvantage: *the interlace flicker*. This can be avoided if you know what causes it, and the solution is rather simple.

PUTTING IT ALL TOGETHER: PICTURES

The graphics modes with 512 (400 on NTSC) pixels vertically use *interlacing* to double the vertical resolution. This is a technique where the monitor displays pixels between the regular 256, which makes it take twice the time to redraw the whole screen. Imagine that the pixels from top to bottom are numbered from one to 512. The first time the electron beam in the monitor or TV refreshes the screen, the pixels with even numbers (2,4,6,8 and so on) are drawn. The next time, the pixels with odd numbers (1,3,5,7...) are drawn. This means that every second line is refreshed each time, and the result is that the picture seems to flicker. A normal TV picture is always interlaced, but since there are lots of colors that blend into each other, the effect is seldom noticeable.

If you draw a one-pixel horizontal line in an interlaced resolution, this line will seem to jump up and down. And thin, horizontal lines are the absolutely worst thing you can have in your pictures. A two pixel line appears much more steady.

The Amiga 3000 has a built-in display enhancer that doubles the scan rate to avoid flicker. This requires a VGA or multisync monitor as output, and works well for presentations that are supposed to run on such screens. But if you are going to tape your script to video, you are restricted to the standard resolution of the video signal. Then you have to take certain precautions to avoid the flicker.

Never use thin, horizontal lines in colors that are in sharp contrast with the background (like white on black). Deluxe Paint IV has built-in functions for antialiasing and smoothing, and these may also help you improve the quality. If you use a regular Amiga monitor when you are painting your pictures, you will see if things start to shake. But if you use a VGA monitor to make a script that is to be taped to video, unpleasant surprises may occur. Always check the result on a video monitor!

PUTTING IT ALL TOGETHER: PICTURES

Painting

You can make your own background images in any Amiga paint program. The most popular one is undoubtedly Deluxe Paint IV by Electronic Arts. This program lets you paint and animate in both low and high resolution, and even in HAM mode.

In addition to the normal painting functions like line, circle and fill, you can also use perspective, shading, transparency, color cycling and other exciting features. Some of the Scala MultiMedia backgrounds are made with Deluxe Paint IV.

We recommend that you make your backgrounds in the highest resolution if you are going to add text in Scala MultiMedia. If you use the lowest resolution, they will become chunky.

Digitizing video

There are a large number of video digitizers available for the Amiga. These let you capture video images and convert them to one of the Amiga graphic resolutions.

There are different methods of doing this, but the final result depends very much on the quality of the video signal.

Some solutions, like the DCTV from Digital Creations, can digitize and display images with more colors than the Amiga itself, and such images can currently be displayed, but not texted, in Scala MultiMedia.

As new graphics boards become available, EX modules that support these boards can be developed.

Scanning

If you use photos in your presentations, the results will be best if you use a flatbed color scanner to read them into the computer. Both EPSON and Sharp have color scanners that

PUTTING IT ALL TOGETHER: PICTURES

can be used with the Amiga, and they are suitable for anything from scanning single-color logos to color photographs. The scanner software comes as a loader module for ASDG's Art Department Professional, and you can generate pictures in all possible resolutions.

Converting pictures from other computers

If the pictures you need are already available on another computer, like an IBM-compatible PC or a Mac, it's quite easy to move them to the Amiga and convert them to IFF format.

If the picture is on a PC, you can move the picture to the Amiga by formatting a 720K disk on the PC and copying the picture to this disk. There are many Amiga utilities available for reading such disks, also in Public Domain, and new versions of the Amiga operating system will also be able to do this.

Once the file is on the Amiga, you need to convert it to IFF format. Art Department Professional from ASDG can read and write a number of formats, such as PCX, GIF and TIF. This program can convert any picture to any screen resolution.

PUTTING IT ALL TOGETHER: PICTURES

Animations

What is an ANIM?

If you made a series of pictures in a paint program, and showed them at a rate of 25 pictures per second, the eye would see the sequence as a constant motion, just like a regular cartoon film. But since each picture takes a lot of memory, this would hardly be possible.

Some years ago, the so called ANIM format was developed. This is a way to save a series of pictures using *delta compression*, which means that only the differences between the pictures are saved. The ANIM file contains the first picture, plus lots of small squares containing the parts of the picture that have changed for each frame. Since large parts of the screen in an animation often don't move (e.g. a small character moving across a background), this saves a lot of memory. Only the parts of the picture containing the moving character will be saved.

When an animation is played, the parts of the picture that have changed are copied onto the screen. The Amiga graphics hardware has a "blitter", which is a device that is very good at copying rectangular memory blocks, and this is part of the secret behind the speed of the Amiga graphics.

PUTTING IT ALL TOGETHER: ANIM

Making movies

There are many animation programs for the Amiga, both 2- and 3-dimensional. For normal use, Deluxe Paint IV is a powerful and affordable package with lots of interesting possibilities. In addition to simple two-dimensional animation tools, you can also move flat objects along three-dimensional paths. Making rotating logos and text is quite easy.

For those who prefer to make "classic" Disney-style animation, Deluxe Paint IV also offers a "light table". This feature makes it possible to see frames before and/or after the one you are currently painting, just as if the drawings were painted on transparent foils.

3-dimensional

If you want to create dazzling 3-dimensional animations, there are a number of 3-D modelling and animation packages available (Real 3D, Sculpt 4D, Imagine etc.). All of them require a bit more "brains" than Deluxe Paint, and it often takes a long time to calculate the images. An accelerator board is recommended!

One 3-D program that is simple to use is *Scenery Animator* from Natural Graphics. With this affordable package, you can create amazing "flight movies" over a 3-dimensional landscape with lakes, mountains, snow and even trees!

Real-time video

To digitize video, you need - of course - a video digitizer. The best way to do it, is to digitize on a frame by frame basis. Not all digitizers are equally suited for this.

To digitize video with high quality, you need a source with high quality. Since it is a still picture that is digitized, it is important that the frozen video image is good. To avoid flicker and noise, it is preferable to connect a time-base corrector between the player and the digitizer.

PUTTING IT ALL TOGETHER: ANIM

How the digitizing itself is done, depends on which digitizer you use. If the digitizer software can be controlled from ARexx, and you also have software and an interface to control the video player, you could write an ARexx script that does the job for you. Remember that the frames have to be compressed into an animation.

For "home" use, an affordable solution is "The Complete Color Solution" from Rombo systems in England. In addition to digitizing color images, this digitizer is also capable of grabbing two frames per second in black and white. If your video player has a "slow play" mode, you can digitize while the video is played. The result is not 100% perfect, but the results can often be amazing! Remember that a digitized video can be loaded into Deluxe Paint and treated like a regular animation.

Sound and music

What is sound?

The Amiga computer has four built-in sound generators, divided in two stereo channels. Due to the advanced hardware design and true multitasking environment, these sound channels can reproduce sampled sound without slowing down the rest of the system.

The Amiga sound chip lets you add sound to your scripts without buying lots of extra hardware and software. The internal sound chip currently offers only eight-bit resolution, which may not be sufficient for the professionals. For better sound quality, use an external MIDI sampler and control it with the MIDI EX, or control a Commodore CDTV with the CDTV EX.

Scala MultiMedia supports the most common file formats, *IFF sample* for playing "raw" sampled sound, and SMUS, Soundtracker and DSS modules for playing songs consisting of several instruments.

Samples

By using a sampler (such as the GVP Digital Sound Studio), you can record any sound digitally into the Amiga, in a way similar to the dedicated MIDI sampling keyboards often used by musicians. The sound quality depends on the *sample rate* of the sound, that is, how many times per second the analogue source sound is transferred into digital information. The higher the sample rate, the better the sound quality. And of course, the space required to store the

PUTTING IT ALL TOGETHER: SOUND

sound will also increase proportionally.

The maximum sample rate of the Amiga DMA chips is 28000 samples per second. Since the sound has an eight bit resolution (eight bits equals one byte), this means that storing a second of sound may take as much as 28 KBytes!

Some Amiga samplers can sample up to 56000 samples/second on an accelerated Amiga, but playing this kind of sound draws heavily on the CPU, and will not work at all in the multitasking environment of Scala MultiMedia. For example, the sampler software that uses this sample rate often turns off the screen display when playing a sound, which is impractical for Scala MultiMedia.

However, sound quality can be improved a lot even with a lower sample rate by using a good sound source, like a Compact Disc instead of a third-generation tape copy. Also be sure to adjust the input volume to an optimal value. Too low will result in background "white noise", while too high will overload the sampler and produce noise in the peaks.

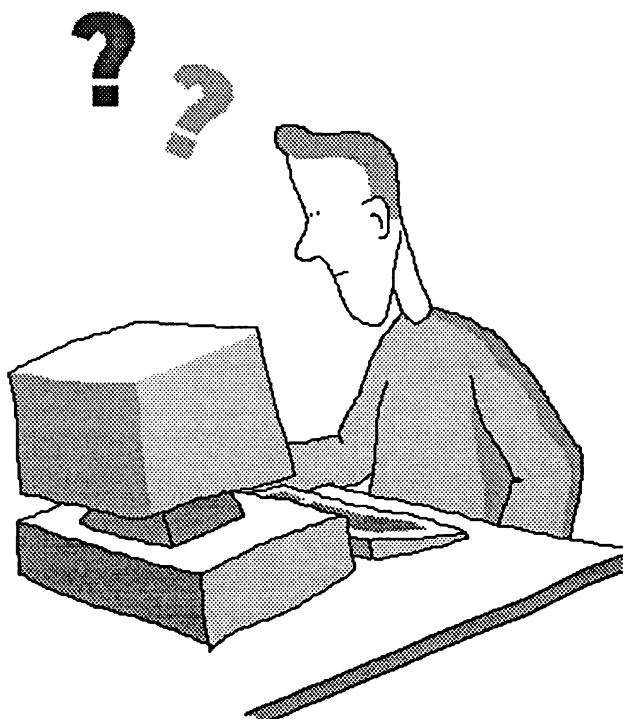
Soundtracker/DSS Modules

These are melodies produced with a special kind of sequencer software available for the Amiga. A sequencer lets you build 4-track music by using sampled sound. The sounds are usually sampled instruments, and the software can transpose the sounds up and down the scale to produce different tunes and harmonies. Since each sample is used again and again, the song can be much, much longer than a "raw" IFF sample without taking up so much space.

You can save your composition as a single file containing the composition *and* the instruments - and this file is called a *module*. The size of the file depends mainly on the number of instruments, and not the length of the song. There are many such modules available, covering all possible music styles - from classic to heavy metal.

SMUS files

Some Amiga music programs support the SMUS format. Scala MultiMedia can also play back these songs, and the instruments that are used must be present in a drawer that has the logical device "Instruments:" assigned to it.



Reference

In this section you will find detailed information on all the menus, buttons and powerful features of Scala MultiMedia.

REFERENCE: OVERVIEW

About this reference section

In this section you will find general information about the basic principles of Scala MultiMedia, as well as detailed information on all the menus and buttons. There is also a large section about Scala Lingo and ARexx.

Where is it?

Here is a "local" index to the reference section:

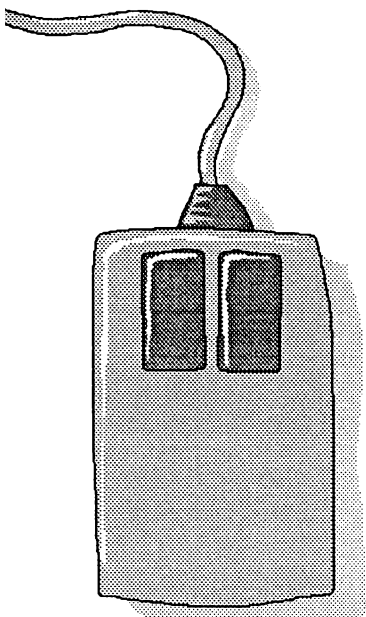
General	122
Main menu	126
File menu	146
System menu	152
Edit menu	158
Font menu	168
Color palette	172
Layout menu	176
Text wipe menu	184
Page control menu	190
List menu	192
Button menu	196
Animation menu	202
Execute menu	206
Variables menu	208
Sound menu	212
Scala Lingo and ARexx	218

REFERENCE: OVERVIEW

This page is intentionally left blank

REFERENCE: GENERAL

The Amiga computer and Scala MultiMedia are mainly controlled with a mouse. This is the little "creature" with two buttons and a long lead connected to the Amiga. Move the pointer on the screen by sliding the mouse across the table. The mouse has two buttons:



Left Mouse Button

This is used to select buttons, text and symbols on the screen. This button is often called the *select button*.

Right Mouse Button

is used in Scala MultiMedia to turn the *Edit* menu on and off. Press the button once to turn off the text menu, press it again and the *Edit* menu reappears. This button also speeds up the changing values when you press the increase/ decrease arrows in the menus, and it aborts move operations in the *Edit*, *Main* and *List* menus.

While running your script you may use the mouse as a remote control, to change pages the same way you would change slides with a slide projector. The right button takes you forward, while the left button takes you back one page.

In an interactive script with buttons, either mouse buttons can be used to select.

If you press both mouse buttons on this arrow, the value will decrease rapidly.

Speed 5

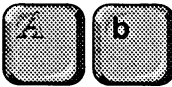


*This is the right
Amiga-key*

The keyboard

Even if the mouse and the menus make Scala MultiMedia easy to learn and simple to use, some functions may be more convenient to execute directly from the keyboard.

The top row of your keyboard consists of 10 function keys marked F1 through F10. In addition to these a number of "Amiga-key combinations" are available. The keyboard has two Amiga keys, one on each side of the spacebar. Certain functions may be performed by pressing the right Amiga key in combination with another key.



Makes the text bold

Throughout this manual the keyboard shortcuts are indicated by small symbols like the one to the left. This one you will find in the text style paragraph in the *Main* menu reference section. It means that if you hold down the right *A* key, and then press *b*, the line you are currently editing will be bold.

You will find all the available keyboard shortcuts on The Quick Reference Card at the end of the manual, and in appendix C.



Stops a script

A final key word

The *ESCAPE* key is situated at the top left corner of the keyboard. Use this key to stop the script and return to the menu. If you hold down the *SHIFT* key while pressing *ESCAPE*, Scala MultiMedia will go to the *Edit* menu, and you can edit the page that was displayed when you pressed *SHIFT+ESCAPE*.

The *ESCAPE* key also serves as *No* or *Cancel* when Scala MultiMedia displays a message and wants you to respond before proceeding. The *RETURN* key (large key with an angled arrow pointing down and left) is equivalent with *Yes* or *OK*.

Common elements

Scala MultiMedia has a series of menus. Among these are the *Main* menu, the *System* menu, the *File* menu and the *Edit* Menu. These will be discussed in more detail later in this chapter. You will also see message boxes from Scala MultiMedia when the program needs to ask you for additional information or confirmation. The menus contain some common elements. These are:

Buttons



A *button* is a rectangular area on the screen which looks as though it can be "depressed". You select these buttons by clicking on them. The buttons are used to turn things on and off or to start a process.

On/Off Buttons



If you for instance want to add shadow to a text, select the *Shadow* button in the *Text* menu. This button will then remain "in", which means that the shadow is turned on. In this case, Scala MultiMedia illustrates it by simulating that it is depressed.



In most cases, buttons that are turned *on* are marked with a checkmark behind the button name.

When you click these buttons, they will alternate between on and off.

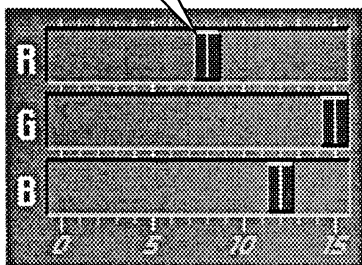
Action Buttons



Other buttons start a process. If you for example press the *Wipe* button for a picture from the *Main* menu, a menu with the various forms of transitions available will appear. Examples of this kind of button are the *OK* and *Cancel* buttons.



Drag this button left or right to change the value



Value buttons

These buttons are used to set a value, and they consist of two arrows and a value indicator. The arrow on the right side increases the value, while the one on the left side decreases the value. One click on the left mouse button will increase or decrease the value by one. If you hold the button down the value will continue to increase or decrease until you release the button.

Some value buttons consist of two components, *seconds* and *frames*. As default, this value is incremented or decremented by one second. To modify the number of frames, click on the value, and then use the arrows to modify the value. The part that is changed, is written in yellow.

To speed up the rate with which the numbers change, you can press the right mouse button at the same time.

It is also possible to type in a value by clicking between the arrows. A cursor will then appear, and you can type in a new value.

Sliders

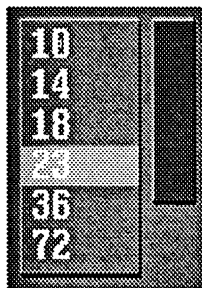
In order to get the right color or set the volume, use a slide control. It works in the same way as the ones you may have on your stereo. It consists of a button that can be moved horizontally or vertically - called a *slider*. Place the pointer on the slider, keep the left mouse button depressed and move (drag) it to its new position. For example, in the *Color Palette* there are three sliders, one for each of the colors red, green, and blue.

REFERENCE: GENERAL



Switch buttons

These buttons are used to cycle through a series of options. The selected option is displayed on the button, and you change it by selecting the left or right arrow.



Lists

When you select a picture or a font, a list will appear which you can leaf through using the slider. The size of the slider button changes according to how much of the list shows on the screen. If only half the list shows, the slider will be half as long as the visible list. If all the values or names in the list are visible, the slide control will be inactive.



The DEL key deletes the character to the right of the cursor, while Backspace deletes the one to the right.

Text Field

When you need to write text in a menu, for example to give a page or script a name, you will be asked to write the name in a rectangular field. Type the name using the keyboard.

You can use the *DEL* or *BACKSPACE* key to delete characters, and the *left/right* cursor keys to move the cursor.



Clears a text field

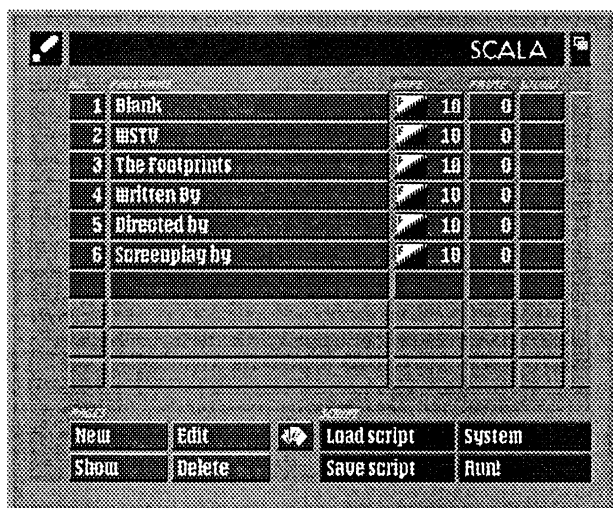
To clear the text where the cursor is positioned, press the right *Amiga*-key and the *x* key simultaneously.



OK/Cancel

You will generally find these buttons in each menu. Use them to confirm that the settings you have made in the menu apply (*OK*), or Cancel the process, and retain the old values you were working with.

The Main menu

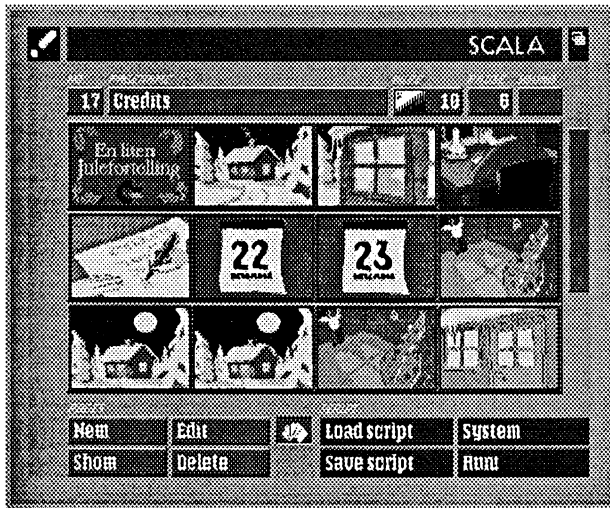


Here you see the main menu with a script consisting of six pages.

The *Main* menu menu is the first to appear when you load Scala MultiMedia. This menu shows a list of the pages in the script, with the transitions, speeds, pauses and EX commands to go with it. From this menu you have access to all the other menus in Scala MultiMedia. You can load a script or build a new one from background pictures, animations, sounds and other events. You can also select to edit a page, and decide in which order the pages in the script will appear.

REFERENCE: MAIN MENU

You can configure the *Main* menu to your suit your own needs and wishes. There are two ways to view the pages.



*This is the
Scala Shuffler!*

The list view vs. the Scala Shuffler

The "traditional" way of viewing and editing a Scala script is to move up and down in the vertical list of names, as mentioned above. In Scala MultiMedia we introduce the revolutionary new Scala Shuffler, that shows you miniatures of your pages. This gives you a more intuitive way to move around in the script and select which pages to edit since the Scala Shuffler will work as a "storyboard" for your presentation.

Switch between the list view and the Scala Shuffler at any time (using the square *Shuffler* button between the *Page* and *Script* buttons). Once activated, the *Shuffler* will keep miniature pictures for all pages in your script, regardless of whether you are using the *Shuffler* or not. To really turn it off, turn off the *Shuffler* button in the *System* menu.



First we will explain the list view. This reference is based on the default configuration of the menu. As mentioned, you can configure the *Main* menu to suit your own needs, but this will be discussed in detail later.

Pages

The main events in Scala MultiMedia are the pages. A page might be a picture with text on, an animation, or anything that Scala MultiMedia can load! When you create new pages, these will appear in the list on the screen. There is room for 10 pages on the screen. If there are more than 10 pages in the script, you use the slider on the right-hand side of the screen to move up and down in the list.

The direct access line

In the List view the last line has a special function. This is the Direct Access Line, which can be used to add events that not include a new picture or animation. For example, let's say that you need to play two different sounds when displaying a picture. Then you would put the first sound in the sound column of the picture, and then have a blank line below, that only contained the second sound (no picture). Then the picture from the previous line will still be displayed.

The Direct Access Line is always located at the bottom of the script, but you can drag it anywhere you want it. When you have used it, a new one will appear at the bottom of the script. You can also use the *Page control* menu to insert it after the current page, or press the *SHIFT* key while selecting *New*.

When there are no script in memory, the Direct Access Line is at the top of the list. Otherwise, it is the last empty line. This is why there is always one blank line that you can't delete.

REFERENCE: MAIN MENU

A line in the script may for example look like this:



Page Number

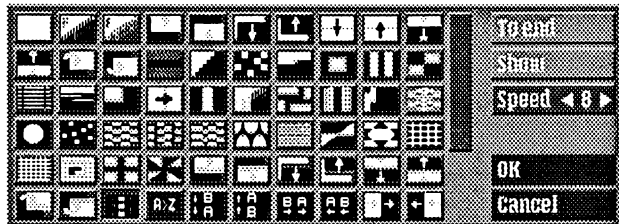
The page number is situated to the left. When you click the page number button, the *Page control* menu will appear.

Page Name

The name of the page is situated to the right of the number. When saving a page, Scala MultiMedia will suggest the first line of text as the page name, but you can alter this. If the page consists of an animation, the original name of the animation will be used. Naturally, it is smart to give the pages names which describe the contents of the page as well as possible. This will make it simpler to go back to the correct page later.

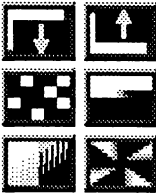
Wipes

Each page can enter the screen in a wide range of different ways. Scala MultiMedia has many page wipes, ranging from simple cuts or fades, to "coin-flips" and rolls. Information about the type of wipe and wipe speed, is displayed on the button to the right of the page name. When you select this symbol, the following menu appears:



Here you see a number of buttons which represent the various types of page wipes available. There are more wipes than can be shown at once, so you have to use the

slider on the right side of the buttons to see all of them. To select a wipe, just click at the button representing it.



We have put a lot of work in making the wipe buttons as descriptive as possible. In the buttons, the white areas represent the new page that is appearing, while the black areas are the previous page. Arrows indicate the direction of movement, and smoothed edges means that the movement is smoothed. The name of the selected wipe is shown in the header.

The speed of the transition can be chosen on the right-hand side. For most transitions, the speed ranges from 1 to 10, but some of the wipes have maximum speed values of 30.

In the top right-hand corner there is a button marked "To end". If this is turned on (indicated by a tick), the selected transition and speed will be applied to all the following pages in this script when you click OK.

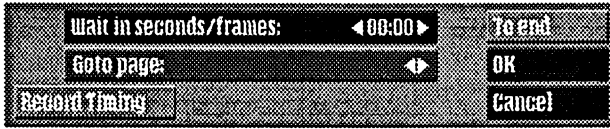
When the *Wipe* menu is displayed, you can change the wipe type for other pages in the script directly, without clicking OK first. Simply click the wipe button for the page you want to change.

To employ the new settings select *OK*, while *Cancel* returns to the *Main* menu without changes. Remember that if you have modified many pages without leaving the *Wipe* menu, only the changes made to the last page will be cancelled.

Pause

This button brings up the *Pause* menu that enables you to select how many seconds and *frames* from one page starts to wipe in until the next page starts to wipe in.. A second is divided in 25 frames on a PAL system, and 30 frames on an NTSC system. If the wipe takes more time to complete, the actual pause will be longer.

REFERENCE: MAIN MENU



Use the *left/right* arrows on the value button to make longer or shorter pauses. If the pause is set to zero, the next page will follow immediately after all wipes has completed. If you try to decrease the value below zero, symbol that looks like a small mouse will appear. This means that when you run the script, the next page will appear when you press the left mouse button.



The button *Record Timing* lets you set the pauses interactively. When you click it, Scala MultiMedia will start showing the script from this page, and proceed to the next page when you press the right mouse button. The time the page was displayed before you pressed the mouse button will be stored in the pause variable for this page. You can continue clicking until you reach the end of the script, or abort the process with the *ESCAPE* key at any time. The pause before you pressed the *ESCAPE* key is not recorded. This is a very powerful way of synchronizing the events to a soundtrack, a videotape, or your own comments.



Note: Be sure to allow wipes to finish before you advance to the next page, or timing will be wrong.

As in the *Wipe* menu, you can click the *Pause* buttons for other pages in the script directly when the *Pause* menu is displayed, without clicking *OK* first. The settings in the wait menu will then change to the ones previously applied to the selected page.

If you want all the following pages to use the same *Wait* settings as this one, turn on the *To End* option before clicking *OK*.

To use the new settings select *OK*, while *Cancel* returns you to the main menu without changes. If you have modified many pages without leaving the *Pause* menu, only the changes made to the last page will be cancelled.

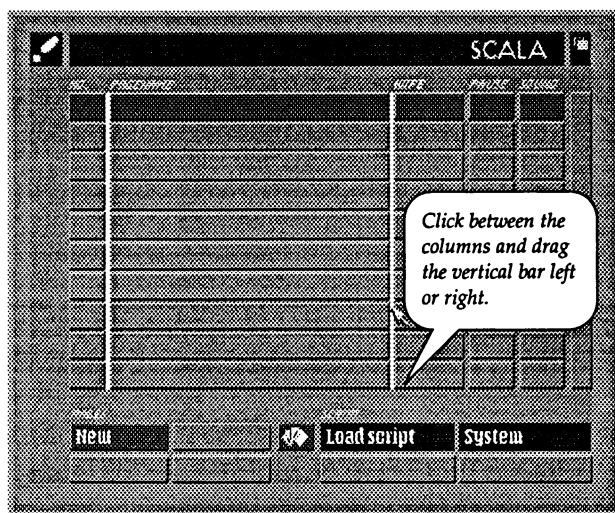
REFERENCE: MAIN MENU

Sound

This button will take you to the *Sound* menu, where you can start playing, stop or change the volume of a IFF sample. SMUS song or Soundtracker/DSS module.

EX columns

If you have installed any EXes in the Scala MultiMedia "Startup" drawer, they may appear in this menu. That is, If the EX does not require input from you, it will probably be "invisible", just working in the background. Otherwise, it will be added on the right side of the sound column.



Configuring the columns

You can change the order and size of the columns in the main menu to suit your own needs and wishes. This may be necessary in order to make all the columns fit on the screen if you use several EXes.

The order can be rearranged using the *System* menu, by moving the buttons indicating Scala EX modules up or down in the EX list.

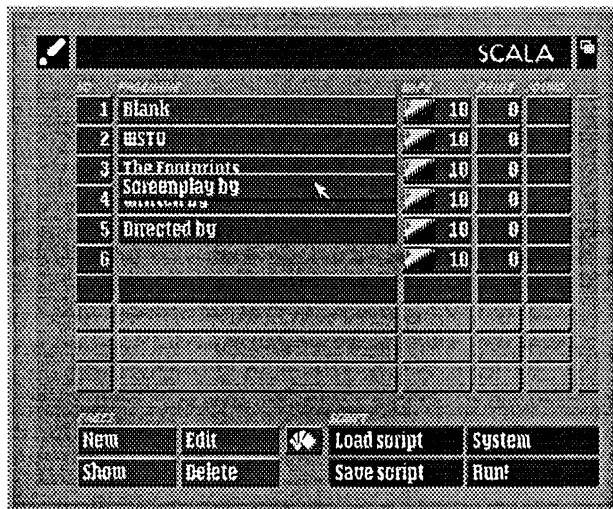
REFERENCE: MAIN MENU

To change the size, simply click to the right of the column you want to change, and a vertical bar will appear. Drag the bar left or right to shrink or expand the column.

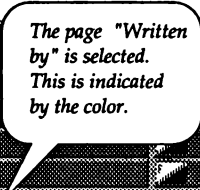
If you have several EXes installed, they will be revealed on the right-hand side when you make the columns smaller.

Moving Pages within the script

In order to move a page up or down in a script, simply point at the page name and keep the left mouse button depressed. While holding the mouse button down, you can move the page up or down the list. Place it in its new position by releasing the mouse button.



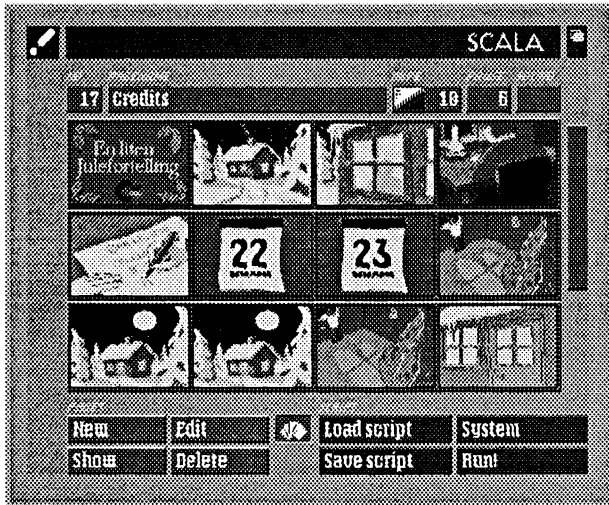
If you want to copy the page or move it outside the visible part of the script, simply click once on the page number. The *Page control menu* will appear.



In the script there will always be at least one *selected page*. When you click on a page name, it is selected. Ypu can see that the name of the selected page is written in another color. When you select a page command (see next paragraph), it is the selected page that is affected.

The script will always contain one blank line at the bottom of the script. This line is called the *Direct Access Line*, and its purpose is to make it easy for you to add new events that don't necessarily contain a picture or an animation. When you want to load a new picture or animation, the fastest way to do this is to click the *New* command, but an alternative way is to double-click the page name button on the *Direct Access Line*.

REFERENCE: MAIN MENU



The Scala Shuffler

This is an alternative way of viewing your scripts, based on miniature images of the pages. You can select how many pictures you want to see (this is done in the *System* menu), ranging from one to 14 per row. Above the shuffler itself, you will see information about the selected page, just as in the normal list view. The miniatures are placed in a left to right, top to bottom order. If there are more pictures than can be shown at once, use the slider to the right to move up or down.

The script line above the shuffler is used in the same way as in the list view. Use it to set wipes, speeds, pauses, sound and other EX commands on the currently selected page.

To move a page, simply drag it to its new position. The other pages will be rearranged automatically. To abort a move, just press the right mousebutton, and the page will jump back to its original position.

Double-clicking a picture in the shuffler is the same as selecting *Show* for this picture.

Page commands

Below the script listing (or shuffler), on the left-hand side, there is a set of commands which apply to pages. When the script is empty, only the *New* command is available, since there are no pages to delete, edit or show.

PAGES



Create a new page.

New

This choice enables you to create a new blank page, or to load a background picture or an animation. The *File* menu will appear, and from this you can select which background picture you want to load. If you press *OK* without selecting a picture file, you will be asked to enter the resolution and the number of colors for a new, blank background page. Then the *Edit* menu will appear.

If you have selected an animation, you will be taken directly to the *Animation* menu.

An interesting possibility is to select a script file when you use the *New* command. The name of the script will then appear on the *Page name* button in the script.

This script will work as a sub-script to the one you are editing. When this line in the script is executed, Scala MultiMedia shows the script once. When the sub-script is finished, the next page will be displayed.

You can for link together a series of different scripts together, by just loading them this way!

This technique is used in the "ScalaMain" demo script, which lets you view the other demo script by just selecting them in a menu. Study this script!

REFERENCE: MAIN MENU

Show

Should you not wish to run the whole script, but only want to show a few pages, the function *Show* is very useful. First select the first page you want to see, and then select *Show*. Scala MultiMedia prompts you to select the last page you want to see, and when you have done this, the selected sequence will be displayed. To see a single picture, double-click *Show*.

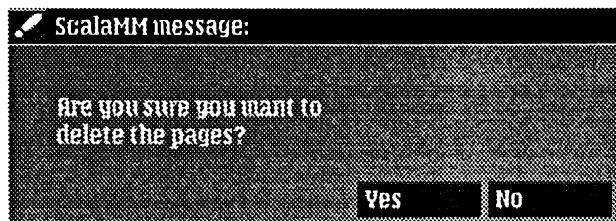
Edit

Should you wish to edit a page, select a page name and then select *Edit*. If the page was an animation, the *Animation* menu will appear. If the page was a sub-script, you will see the *File* menu where you can select to load another script. If the page had a background picture, the text menu and the selected page will appear. If the page had a blank background, the *Blank page* menu appears.

A quicker way to start editing a page, is to simply double-click the name of the page.

Delete

In order to delete one or more pages from the manuscript, use the *Delete* command. Select the first page you want to delete, then click *Delete*. Scala MultiMedia tells you to click on the last page you want to delete, but you can also click *Delete* again to delete the current page.



When you have done this, Scala MultiMedia will ask if you are sure you want to delete these pages, and if you answer OK, the pages will disappear from the script (not from your hard disk or diskette!). If you don't want the pages to disappear, select *Cancel*.

REFERENCE: MAIN MENU

Script commands

To the right of the page commands, you will find commands which apply to the whole script. These are:

SCRIPT

Load script	System
Save script	Run!

Load script

To load a script from a diskette or a hard disk, select *Load Script*. The *File* menu will then appear requesting you to select a script.

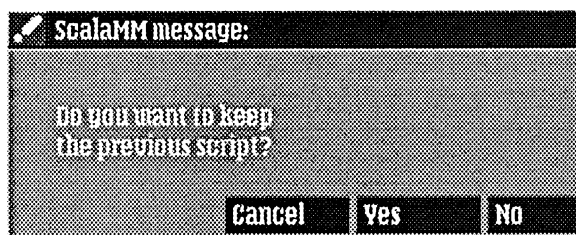
If you are in doubt which name the script has, you can click once on a script name and then select *Show* in the file menu. The selected script will start playing. To stop it, press the *ESCAPE* key. The *File* menu will appear again, and if it was the correct script, click *OK*. If not, try another one!

While previewing a script, you can press *SHIFT+ESCAPE* to select *OK* automatically.

You can load more than one script simultaneously by holding down the *SHIFT* key. The number of selected scripts is displayed in the *File* field. The scripts will be loaded in the order you select them.

When you have selected one or more scripts, select *OK* to load them. See the reference section *File menu* for more information on how to use the *File* menu.

After the script has loaded, you will return to the menu. If you already have a script in memory, you will be asked if you want to add the new script to the old one.



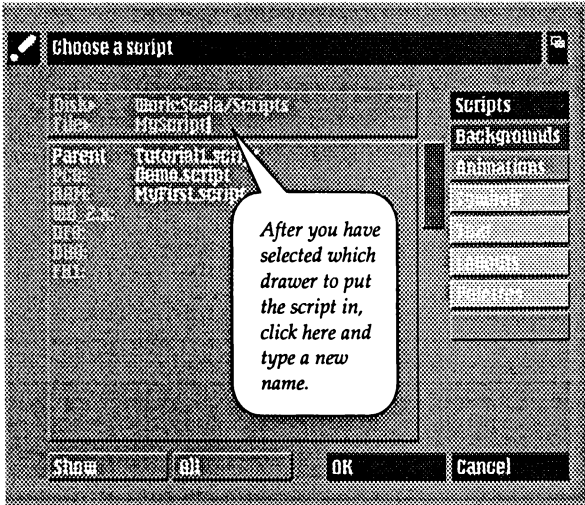
If you select *Yes*, the new script(s) will be inserted after the active line in the previous script. *No* clears the previous script from memory before loading the new one(s), while *Cancel* takes you back to the *Main* menu without loading or deleting anything.

Run!

When you want to run your script, you select (of course) *Run!* The pictures in the script will be shown, and the transitions and pauses you have decided on will be employed.

The keyboard shortcut for *Run!* is to press the space bar.

REFERENCE: FILE MENU

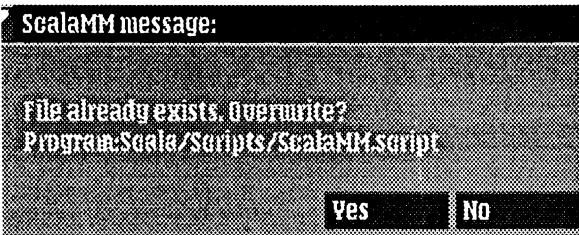


ve script

is command saves the script. If the script you are going to ve is a new one, you have to type the name of the script in e "File:" field in the *File* menu. If you wish to replace an isting script file, click at its name in the file list, and then ect OK.



Saves the script.



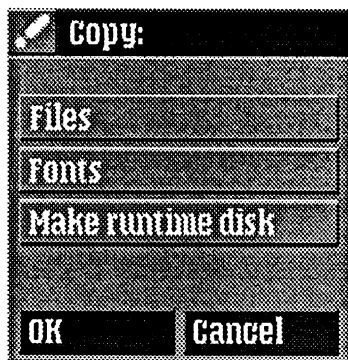
he name already exists, you will be requested to confirm it you really want to replace it.

REFERENCE: MAIN MENU

Copying files

A Scala MultiMedia script is a detailed description of which background pictures that are used, and which fonts and point sizes the text lines are written in. The pictures and fonts themselves do not automatically follow your script if you move the script file to another disk. But Scala MultiMedia can automatically copy all the "extra" files to the specified disk for you, and also make a runtime diskette containing all the necessary files.

When you save a script and all of the files in your script (pictures, animations, sounds etc.) are not located on the same disk or partition, a menu will appear, asking you what you want to move to the disk or partition where you are saving the scripts.. The choices are *Files*, *Fonts* and *Make runtime disk*.



The term *Files* means all files that Scala MultiMedia needs to run the script: pictures, animations, symbols, sounds and music. When the files are moved, new drawers will be created on the target disk with the same names as the original drawers. For example, if the picture "Car" is located in the drawer "Pictures" on your hard disk, a "Pictures" drawer will be created on the target disk, and the "Car" picture will be copied to this drawer.

If you also need to copy the fonts onto the target disk, turn on the *Fonts* option. This will not be necessary if you know that the needed fonts are present in the FONTS: drawer on the computer you are going to run the script on.

Making runtime scripts

If *Make runtime* is activated, the necessary parts of the operating system are also moved to the disk, along with the Scala MultiMedia player module. You need to format the necessary amount of diskettes before you start. The result will be a "runtime disk", and to use this disk you simply insert the disk in any Amiga (with the required amount of memory and a Scala Key). The player will start and start showing the script automatically!

If the script does not fit on one disk, problems may occur when you show the script. If the computer has only one disk drive, while the script is two disks, the ScalaMMPlayer will prompt you to insert the next disk when it is needed.

This would be inconvenient if the script is a looping information program that is supposed to run without attention. In this case you could use two disk drives. However, we recommend that a computer that is used to run scripts continuously over a long period are equipped with a hard disk or enough memory to hold all pages. This will be both faster and more reliable.

Moving scripts to another computer

If you want to move your script to another computer's hard disk, you can also use the *Copy files/fonts* option. and then save the script with *Copy files/fonts* activated. When the disk is full, you will be prompted to insert the next diskette, and so on, until the whole script is saved. Then, move the disks to the other computer, and select *Load Script* from the first disk in the series. Scala MultiMedia will tell you to insert the succeeding diskettes as they are needed. To save the script on the hard disk of this computer, simply select *Save script*, and remember to activate *Copy files/Fonts*. Then, the

REFERENCE: MAIN MENU

whole script will be copied to this computer.

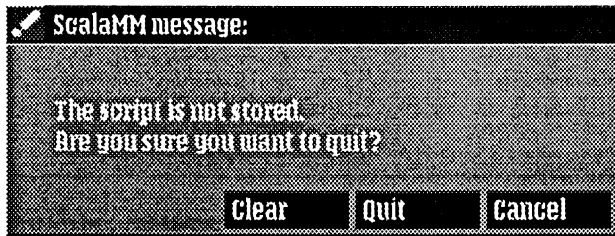
If one or more of the files are so large that they won't fit on one disk, you will run into problems. In these cases we suggest that you use a backup-utility like *Quarterback* or *BRU*. The latter is a part of *Workbench 2.0*, and allows you to make backups to diskettes and tape streamers.

Quit Scala MultiMedia or clear the script

When you click the exclamation mark in the top left-hand corner of the screen, you will be asked if you want to terminate the program, clear the script currently in memory, or return to the program.



Quits Scala Multimedia.



If you select *Clear*, you will return to the *Main* menu with a blank script, while *Quit* returns you to *Workbench*.

Warning: If you select *Clear* or *Quit* without having saved your script first, the script (or all changes since the last save) will be lost. If you are not sure, select *Cancel* and save the script before terminating.



SCALA multitasking

Like all well-behaved Amiga programs, Scala MultiMedia can run simultaneously with other applications.

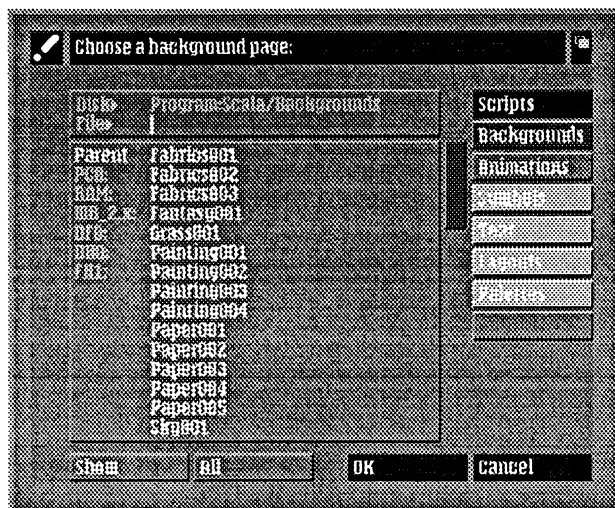
In the top right corner of the screen you find the *front/back* button. When you click this button, the Scala MultiMedia screen will move to the back, and the screen behind will appear, for example the Workbench screen.

Most Amiga programs have a similar button, and you can use it to cycle through all the available screens. If nothing happens when you press the *front/back* button, the Workbench screen may be closed to save memory. Opening and closing the Workbench screen is done from the *System Menu*.

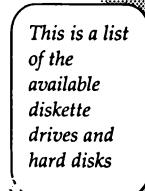
You can also slide the Scala Multimedia *Main* menu down by dragging the title bar at the top.

The File Menu

When you load or save something to or from a diskette or hard disk, the *File menu* will appear.



At the top of the screen Scala MultiMedia tells you what you are about to do. In the middle of the screen there is a list of the files and drawers as well as a list of all the disk drives and hard disks available, like RAM:, DF0: and DH0:



Here you see the files and drawers.

By "drawer" we mean part of a diskette or a hard disk which has been given a name. For example, you can have an drawer named "Pictures" which contains your own background pictures. Please refer to the Amiga user manuals for how to create drawers.

If a drawer contains more files than can be displayed at one time, use the slider to leaf up or down in the list to find the file you want.

REFERENCE: FILE MENU

Above the large frame there is a smaller one where you will find "Disk" and "File", this shows which disk you are on and which file you have chosen.



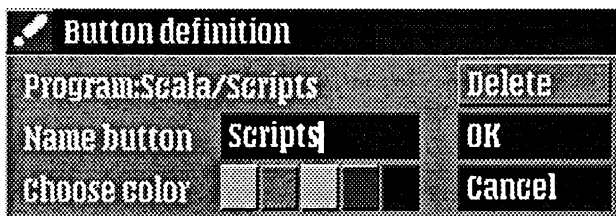
Disk

This is where you can find the disk you want to save or load from. Click on its name to the left of the file list (for example df0: or dh0:), and then select drawers as described earlier. You can also use the "Quick selection" buttons to the right.

Making your own buttons

You can define your own buttons (to the right) for easy selection of disks and drawers. Scala MultiMedia comes with a predefined set of buttons, but you can make your own like this:

- ◆ Using the *File* menu, select the disk and drawer you want the new button to represent. Let's say you have started making your own background pictures with a drawing program, and these are located in a drawer called Work:Graphics/Images. First have to display the contents of this drawer.
- ◆ Notice that the bottom button on the right is blank. If you select this button, you will be asked to enter the name of the button, and decide its color.



REFERENCE: FILE MENU

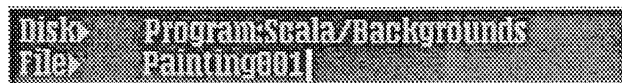
- ◆ When you have done this, click *OK*, and the previously blank button will display the specified name and color. Just below it, a new, blank button appears.

From now on, the button you defined will take you to its drawer no matter where you are when you click it. To edit a previously defined button, simply hold down the *SHIFT* key while you click the button. Then you can change the name and color of the button, and it will be redefined to the current drawer.

Use this feature to customize Scala MultiMedia for your own needs, so you won't have to bother about where directories scripts, pictures and symbols are located. Simply press the correct button!

File

When you select a file, its name will appear after the word *file*. You can also write the name yourself, as you do when selecting a disk.

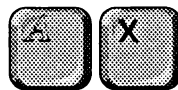


Select a File

When you select a file name, it will change color. At the same time, the name will appear in the space, "File", above.

New file name

To save a file with a new name, click in the *File* field. A cursor appears, and you can type a new name. You can use the *DEL* and *BACKSPACE* key to delete letters, or press the right *Amiga*-key and *X* to clear the field.



Select More than one File

You can load a number of files at the same time. To do this, keep the *SHIFT* key depressed when choosing the files. The files you select will then be marked (the color of the letters will change), and the number of selected files will appear in the *File* field.

All

To select all the files in the *file list*, select *All*. All the names will then be highlighted and the number of files will appear in the *File* field.

Show

In order to check that you are about to load the correct script, picture, animation, sound or music, select the *Show* button in the bottom left corner.

You will then see the selected script, animation or picture, or hear the selected sound or song.

OK

When you have finished selecting files, you must select *OK* to fetch or store. If you have selected a picture, the *Edit* menu will appear.

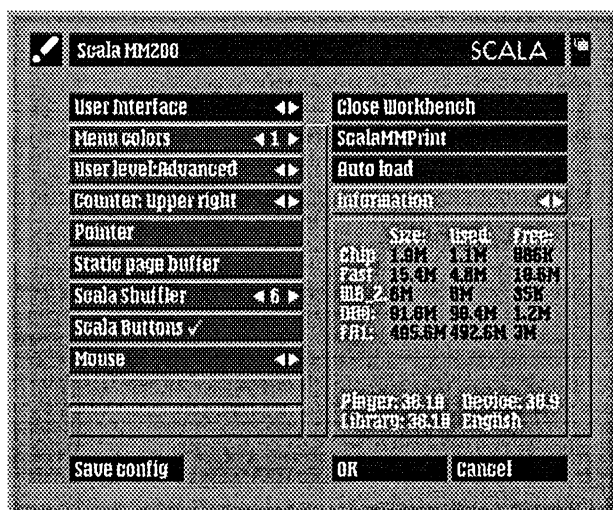
Cancel

If you do not wish to save or load the marked file(s), you can stop the whole operation by pressing *Cancel*.

REFERENCE:FILE MENU

The System Menu

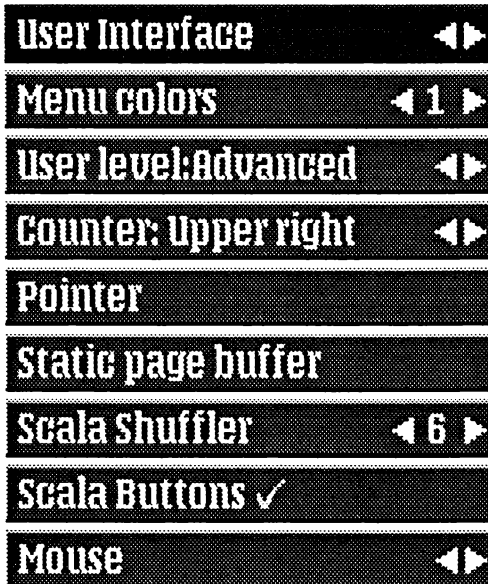
This menu lets you influence the way Scala MultiMedia functions. To get to this menu, select *System* from the *Main* menu.



User Interface / Scala EX

The top left button toggles between two different button sets below. These are *User Interface* and *Scala EX*. First we will explain the *User Interface* buttons.

REFERENCE: SYSTEM MENU



Menu Colors

In order to adjust Scala MultiMedia to your own taste or mood, you can select which color you want the menus to be. There are over 20 different professionally designed color palettes. Try them!

Counter

If you want to see which picture you have reached, you can activate a counter in one of the corners of the picture. This will continuously show the number of the current picture.

Pointer

When you want to run a script it is not certain that you want the pointer to show on the screen. You can choose whether with this button. If the pointer is visible there is a tick after the word *Pointer*. To turn it off or on you press the button once.

REFERENCE: SYSTEM MENU

Static Buffer

When this option is activated, Scala MultiMedia will automatically load all the pictures and animations of a script into memory when the script is loaded. This can speed up playback, since nothing is loaded from disk when the script is running, but it requires that you have as much RAM as the script and all the pictures and animations takes.

Scala Shuffler

Use the increase/decrease arrows to decide how many pictures you want on each line in the Shuffler. The number can range from one to fourteen, which gives you a maximum number of 112 pictures on the screen at once!

When you switch from the list view to the Scala Shuffler in the *Main* menu, Scala MultiMedia will start generating the miniature pictures. If you want the pictures to be generated at once when a script is loaded or new pages are created, turn this option on (indicated by a tickmark).

If you never or seldom use the *Shuffler*, turn this option off.

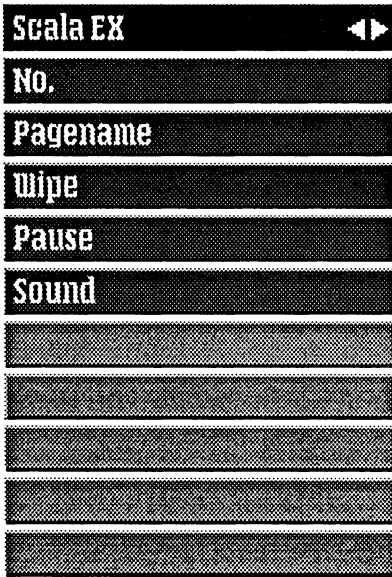
Scala Buttons

If the currently loaded script contains interactive Scala Buttons, this option will appear. If you want to run the script without buttons, turn this option off.

Joystick/Mouse/Numeric keys/Function keys

Scala MultiMedia supports input from both mouse, joystick and keyboard (function keys and numeric keypad) when you run an interactive script. With this button you can cycle through these four options, and turn each of them on or off. See the *Buttons* menu for more information about how they work.

REFERENCE: SYSTEM MENU



The Scala EX column

When the top left button is set to *Scala EX*, the column below will display all the Scala EX modules that are located in the "Startup" drawer.

You can rearrange their order. To move an EX, simply drag it up or down in the list, and drop it wherever you want it. You can move *all* the columns in the *Main* menu this way.

If you have more EX modules than can be shown at once, use the slider to the right to move up and down in the EX list.

For documentation of the EXes, see the end of this chapter.

REFERENCE: SYSTEM MENU

These are the internal EXes:

No.	The number of the page, usually the far left column in the <i>Main</i> menu.
Page name	The name of the page
Wipe	The wipe column
Pause	The pause column
Sound	The sound column
Additional EXes	will be added if they are present in the <i>Startup</i> drawer when you start Scala MultiMedia.

On the right side of the screen

You will find the following buttons on the right side of the screen. They are visible all the time, regardless of which setting the *User Interface / Scala EX* button is set to.

Close Workbench

You can open and close the *Workbench* screen using this button. If another application is running on the *Workbench* screen, you won't be able to close it. If you receive "memory low" warnings, closing the *Workbench* will free some memory.

ScalaMMPrint

This is an utility that can make a hardcopy of your script on a printer. The *ScalaMMPrint* utility is described in the "Utilities" appendix.

Auto Load

When you select *Auto Load*, you tell Scala MultiMedia to load the current script automatically when you start the program. A file called "ScalaMM.startup" is created in the s drawer of the selected device. This file contains the name of the script currently in memory.

REFERENCE: SYSTEM MENU

The Team/System Information

Shows information about who-is-who in the Scala MultiMedia developer team, or information about your computer, such as available memory and the amount of free space on diskettes and hard disks. The Amiga memory is divided in two: CHIP and FAST. Graphics must be stored in CHIP memory, so if this number is too low, you may run into problems.

Save configuration

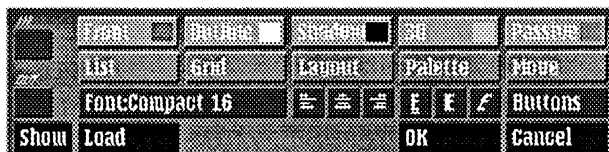
This function stores any specified settings in the file "S:ScalaMM.config" which will automatically be read when you later start up the program.

OK/Cancel

To make the changes apply, select *OK*. *Cancel* will exit to the *Main* menu and "forget" your changes.

The Edit menu

This menu contains all the functions used to add text and brushes on pictures, such as typing text, setting the colors, justifications and textline transitions. The *Edit* menu covers the bottom of the picture, but can be hidden at any time by pressing the right mouse button once. Press the right mouse button once more to get it back.

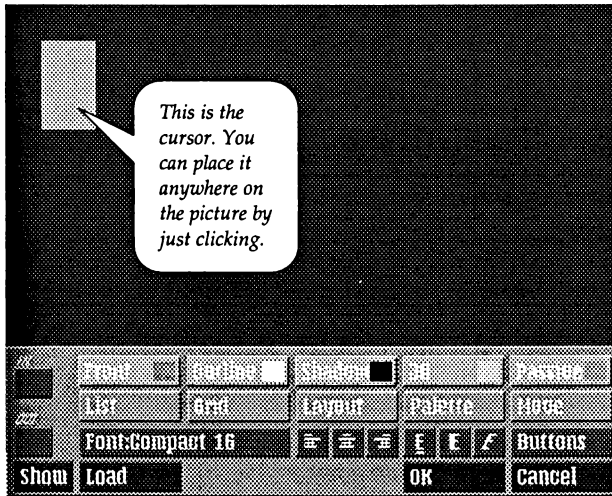


How does Scala handle text?

The *Edit* menu in Scala MultiMedia works as a kind of "graphic word processor". In the same way as with a normal word processor you can edit, insert and move text. There are, however, some major differences which it is important to mention.

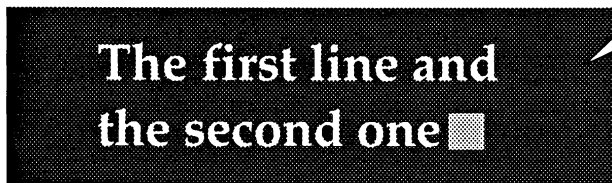
When textung a picture you usually want more control over each line of text than when using a word processor. Therefore Scala MultiMedia deals with each line separately. There are also ways in which to deal with a number of lines at the same time.

REFERENCE: EDIT MENU



Typing text

To write a text in Scala MultiMedia, move the pointer to the desired location on the background picture and click the left mouse button. A *cursor* will then appear where you clicked. The cursor is a rectangle which shows you where the next letter will be written. When you press a key on the keyboard, the letter appears where the cursor was, and the cursor then moves one position to the right.



When you have written a line, press the *RETURN* key to make the cursor jump down and create a new line.

If the Shadow button is pressed, the shadow will only be turned on (or off) for one line.

**The first line and
the second one
and even one more**

Lines

SCALA treats each line of text as a unit. Settings such as colors, typefaces, size, italicization, transitions etc. are selected for a whole line at a time. If you change the color of the text when the cursor is in the middle of a line, the whole line will change color.

To move between the lines you have written, you can either use the keyboard *up/down* cursor keys or simply select the desired line using the mouse. The arrow keys will select lines in the order they were created. To split a line at the cursor position without moving the rest of the line down, press the *ENTER* key.

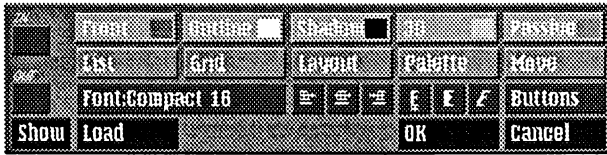
Multiple selection

When you want to change more than one line of text, at the same time, mark the lines with a "rubber band". First move the pointer to the left and above the text you want to change, hold the left mouse button down and move the arrow down towards the right corner. All the lines touched by the square will be affected by the changes you carry out. Remember that Scala MultiMedia treats the whole line as one object, so you will not be able to change the color of half a line in this way. To get rid of the rubber band simply click the picture once. This type of marking can also be used to delete a number of lines at once. Press the *DEL*-key, and the marked lines will disappear.

REFERENCE: EDIT MENU

Keyboard Commands

The experienced Scala Multimedia user will probably prefer to select certain functions directly from the keyboard. This is naturally possible, and in this reference the keyboard commands are found in the margin where applicable. A complete list is found in appendix C and on the *Quick Reference Card*.



Below is an explanation of the buttons in the *Edit* menu. The various sub-menus, such as the *Color palette* and *Layout* menu, are explained later in this section.

Front

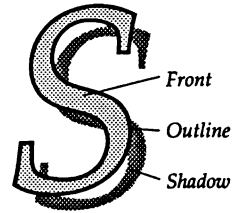
This button selects the front color of the letters. To choose a color, first click at the color you want (from the palette just above the text menu), and then at the small colored square in the button. The line where the cursor is will then change color.

Outline

This button controls whether or not the letter should have an outline, and if so, which color it should be. To turn on outline, click the *Outline* button once (not the colored square). If the cursor is in a line of text, the whole line will be outlined. To turn it off you give one more click. The outline color is selected by clicking at the desired color in the palette, and then in the colored square, in exactly the same way as we did when we selected the front color.

Shadow

This button controls whether or not the letter has a shadow, and the color of the shadow. Both activation and choice of color are controlled in the same way as for outline.



Click here to set the outline color to the currently selected color..



...and here to turn on the outline.

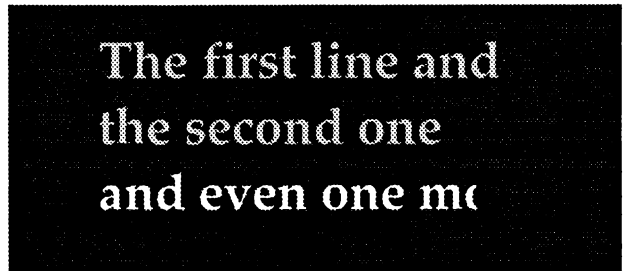
3D

This symbol decides whether the letter should have a 3-dimensional appearance. This is done by drawing the letter in a special way with two colors, one on each side. Select two colors of the same shade, but where one is a little darker than the other. A very stylish effect, but when used incorrectly it can make your text difficult to read (for example yellow letters with green and pink 3D on a red background). Use it in a way that makes it look realistic!

Passive

If you want a text line to change color when the next line is about to appear, activate this button and select which color you want the text to change to. This can be a useful feature if you want to "highlight" lines as they appear.

When the Inactive function is turned on, the line will change to the color indicated on the Inactive button when the next one starts to wipe in. In this example, the third line is about to wipe in. The other two have changed to the Inactive color.



List

This is where you can see a list of what is going to happen when this picture appears, showing in which order the lines will appear, and the selected text wipes and pauses See *List Menu*.

Grid

This enables you to turn an invisible grid on and off; Lines of text will lock onto the grid when moved. This makes it easier to align the text lines manually. The size of the grid can be set in the *Layout menu*.

REFERENCE: EDIT MENU

Layout

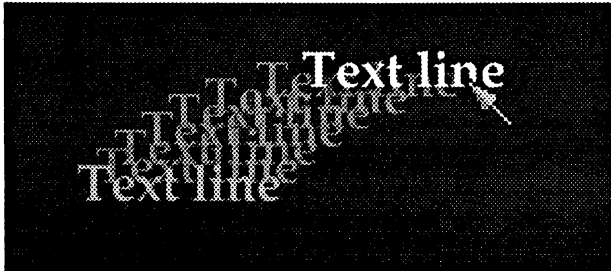
This choice gives you a menu which lets you control the margins, tabulators, line splitting, shadows etc. See *Layout Menu*.

Palette

This gives you the opportunity to change the pictures colors. See *Color palette*.

Move

To move one or more lines of text on the screen you select *Move*. The text menu disappears and you can move the lines by pointing at them and holding the left mouse button down. The line will then "stick" to the arrow and appears as a shadow. Move it to where you want it and release the button.



When you have selected Move, you can drag the lines freely around the screen. However, if one of the justifications are turned on turned on (see below), you will only be able to move the text up and down.

To abort a move, click the right mouse button before releasing the right one, and the text jumps back to its original position.

Font

This is where you select the size and kind of font (typeface) you want the text written in. The name of the chosen font and its size are shown in the button. See *Font menu*.

Justification

Each line of text can automatically be aligned in 3 ways: Left justified, centered or right justified. Of course, only one of these can be active at one time.



From left to right: left, center and right justification

Text Style

The text can be underlined, bold or italicized. All the three attributes can be combined freely. The degree of italicization, thickness of the underlining and boldness can be adjusted for each line in the Layout Menu.



From left to right: underline, bold and italic

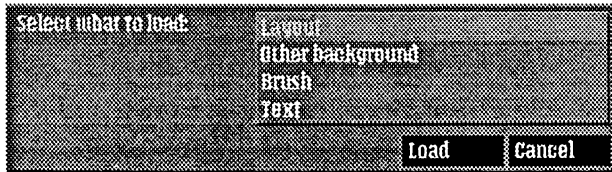
Show

To see the picture with the selected text wipes you can select *Show*. Scala MultiMedia will also play sounds and perform other EX-related actions that are set for this page in the *Main* menu.

REFERENCE: EDIT MENU

Load

The *Load* menu allows you to load the following things:



Layout Loads a Layout which has been saved in the *Layout* menu.

Other

Background If you want another background picture, select *Other Background*. The *File* menu will then appear, and you can choose another background. This background will appear on the screen with the text from the original page.

Brush

Brushes are small pictures created in a paint program (Deluxe Paint). A brush, for example a logotype, can be loaded using this command. A brush is treated in the same way as a line of text. You cannot alter the size, but you can move it around freely. If it is a single-color brush, you can select front color in the same way as with a text. You can also use shadow, justification, etc.

Text

You can write the text in a word processor, and load it by using *Load Text*. The text will flow out on the page. If there is not room for all of it, Scala MultiMedia will load a new background of the same type and continue there. It will keep doing this until all the text has been loaded.

REFERENCE: EDIT MENU

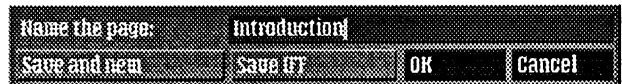
It may be an advantage to turn on *Margins* and *Word Wrap* in the *Layout* menu, so that Scala MultiMedia automatically splits the lines as necessary.

The text must be "clean", in other words in a "generic" (ASCII) format. Most word processor programs are able to save text in this format.

It is possible to control the creation of new pages by using the command NEWPAGE in the text file. Whenever this text, written in upper-case, appears, Scala MultiMedia will create a new page. This makes it easy for you to make headlines for each page - just add NEWPAGE on the line before the headline!

OK

When you have finished a page select *OK*. You will then be asked what the page is to be called, and Scala MultiMedia will suggest the first line of text as the name. Should you not wish to use this name, erase it and write another.



There is also an option called *Save and New*, which saves the page and creates a new one with the same background and layout. This is convenient when you for example are producing a series of almost identical video titles.

The *Save IFF* option saves an IFF picture which contains both the background picture and the text and symbols. This picture can later be loaded into a paint program, or used as a background in Scala MultiMedia.

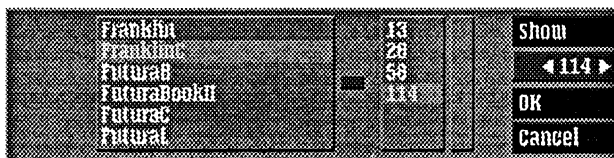
Cancel

If you do not want to save the page, select *Cancel*.

REFERENCE: EDIT MENU

The Font Menu

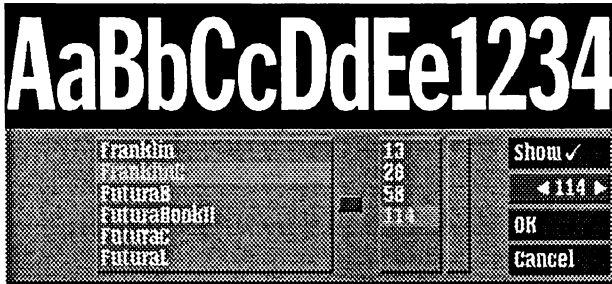
This menu appears when you select *Font* from the *Edit* menu. You can now decide which kind of font you want to use.



On the left you have a list of all the available typefaces, and to the right of the names you can see which sizes the selected font comes in. You can leaf up or down in both lists to find a font and size you like.

To start using a new font, just place the cursor where you want the text to start, activate the *Font* menu, and select the desired font. To change the typeface on an existing line of text, move the cursor to this line by using the cursor keys or by clicking on it, and select another font.

To change typeface on more than one line, use the "rubberband" to select the lines, and select a new font. All text lines that are inside or touched by the dotted square will change to the new font.



Select *Show* to view a sample of the chosen font just above the menu. To employ the new font choose *OK*, while *Cancel* will retain the old values.

Scalable font support

Scala MultiMedia has full support for the scalable "Bullet" fonts found in Workbench 2.04. This means that if you select a font of this kind, you can use the increase/decrease arrows on the *Font size* button to select exactly the size you want. The operating system will calculate a new bitmap font based on the vectorized definition. The quality of the new font size will be optimal, without the "jaggies" that appear when you try to enlarge a standard bitmap font.



Here you see the difference between a scaled vector 128 point Times font and an 18 point bitmap font scaled to the same size.

The point sizes in the size list are the ones created with the *Fountain* utility that comes with Workbench 2.04. This program can also generate bitmap fonts, which can be a great advantage. Calculating the fonts "on the fly" while the script is running requires a lot of time and resources, so if you want optimal speed - use *Fountain* to generate bitmaps in the required sizes.

A hint for those who need lots of scalable fonts: *Fountain* can convert the Compugraphic fonts used by Gold Disk's Professional Page and PageSetter II to the Workbench 2.04 "Bullet" format. A wide range of fonts is available, and your nearest Gold Disk dealer will give you more information about these.

Colorfont support

A wide range of *Colorfonts* are available for the Amiga. These are typefaces where the letters are painted with more than one color, to simulate wood, metal, bricks and other materials.

You can use these fonts in Scala MultiMedia, in almost the same way as single-color fonts. Of course, you cannot directly set the *Font* color, since the letters contain many colors. To change the color of a colorfont, use the *Color palette* to adjust the color range that is used by the font.

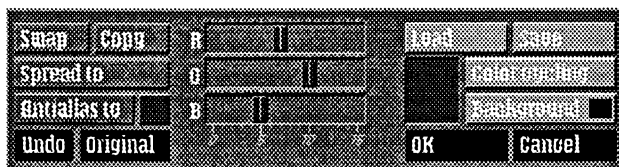
Workbench version 2.04 and higher has built-in support for colorfonts. If you use Workbench 1.3, you need to run a program called "Colorfonts" before you can use these typefaces. This can be easily done by just moving the "Colorfonts" program icon to the Startup-drawer. Then this program will be run automatically when you run Scala MultiMedia.

The *Remap* option in the *Layout* menu may be helpful when you use colorfonts. The colorfonts often use a continuous range of colors in the *Color palette*. If some of the colors in your palette is difficult to change because they are used in a background picture, Scala MultiMedia can automatically find other colors in the palette that match the original ones. Turn the *Remap* option on if you want Scala MultiMedia to pick new colors that match the original ones.

Kara Graphics has released a series of colorfonts called "Kara Fonts".

The Color palette

This menu appears when you select *Palette* from the *Edit* menu. Use it to change the colors of the picture.



Pictures can have from 2 to 64 colors available, depending on the resolution. High resolution images (640 pixels wide) may have up to 16 available colors. These 16 colors can be selected from all the 4096 colors available on the Amiga.

A color consists of red, green and blue components (RGB), and each of these three components may have sixteen different intensities. If all three are zero, the color is black, and if all three are 16, you get white. To change a color first point at the desired color in the palette above, or click at a place in the picture which has that color.

The three sliders in the middle of the menu will then change to match the values of the selected color, and you can alter the color by using the sliders.

REFERENCE: COLOR PALETTE

Copy

This choice copies a color from one area of the palette to another. Click first at the original color, then at *Copy*, and finally at the place in the palette you wish to copy the color to.

Spread

You can make gradual color changes by selecting *Spread to....* First click at a color in the palette, then at *Spread to...*, and finally at the other color. All the colors between these two will be altered to shades which naturally come between the two selected colors.

Swap

This choice interchanges two colors on the palette with each other. First click at one of the colors, then at *Swap...*, and then at the other color.

Antialias to

The *Antialias to* setting in the *Layout* menu smooths the edges of the letters by making an outline of pixels in color hues that are a mix between the letter color and another color. In the *Antialias to* color field, you can set which color the antialiasing algorithms shall use as background color. This setting can affect the quality of the text a lot if you write antialiased text on a multi-colored background. The color you set here, should be equal or close to the dominating background color.

Color cycling

This is a technique often used to simulate motion by rotating a range of colors in the palette. If you have defined color cycling for a picture or animation in a paint program (like Deluxe Paint), this color cycling can be activated in Scala MultiMedia. This option is only available if the background picture contains color cycling information.

REFERENCE: COLOR PALETTE

Background

When you create a blank background, use this button to fill the background with another color than the one on the far left of the palette (usually black). Select a color in the palette, and then select this button.

Undo

Should you regret what you have done, you can select *Undo*. The colors will then change back to the way it was when you entered the palette.

Original

This button resets the palette to the original palette of the background picture. By "original" palette, we mean the colors that belong to the picture.

Load palette

The backgrounds in the Scala MultiMedia Art Library have palettes that are constructed in the same way. This means that you can load palettes from other pictures to make new and exciting effects. When you select *Load palette*, the *File menu* will appear. Just select the name of the background picture or palette file (see below) you want to load the palette from, and the colors will change! You can use *Show* in the *File menu* to check the result.

If you are not pleased with the new colors after the palette has been loaded, select *Undo* or *Original*.

Save palette

Saves the current palette. You must specify a file name.

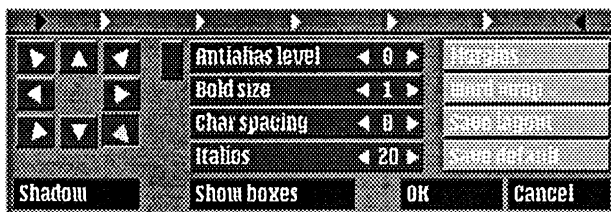
To use the new palette you select *OK*, while *Cancel* exits the menu and resets the palette to its original state.

REFERENCE: COLOR PALETTE

The Layout menu

To make this appear, you select *Layout* in the *Edit Menu*. This menu controls margins and shadow directions. It also lets you save your layout to a file for later use. This feature makes it easier for you to create consistent presentations in a small amount of time.

This menu is meant for the advanced users who needs greater flexibility and control. Here you see the layout menu:



Margins

At the top of the menu there is a field as wide as the screen, containing a row of black and white triangles. The black triangles represent the margins of the page, while the white ones are the tabulators. In order to move a margin or a tabulator, simply drag it where you want it.

REFERENCE: LAYOUT MENU

Antialias level	◀ 0 ▶
Bold size	◀ 1 ▶
Char spacing	◀ 0 ▶
Italics	◀ 20 ▶
Line spacing	◀ 5 ▶
Net size	◀ 6 ▶
Outline size	◀ 2 ▶
Remap	◀ 1 ▶
Shadow length	◀ 4 ▶
Underline air	◀ 2 ▶
Underline pos	◀ 53 ▶
Underline size	◀ 7 ▶
3D length	◀ 6 ▶

In the middle of the screen there is a list of parameters you can set. Use the slide control to the left of the list to leaf up and down. The number after the parameter name is increased or decreased by using the arrows. The choices are as follows (next page:)



Antialias level

Scala MultiMedia has unique real-time antialiasing. The term "antialiasing" means that the software removes "jaggies" from the letters by adding pixels in colors that lie between the letter color and the background color. This gives an impression of a higher resolution, and may enhance the quality of the letters dramatically.

Let's say you have white text on a black background (see left). The antialiasing algorithms look for levels of grey in the color palette, and add pixels around the edges to smooth out the steps. There are four levels, where zero means no antialiasing (top) and three means maximum antialiasing (bottom). The result of the antialiasing depends on the availability of in-between colors in the palette, and that the "antialiase to" color in the palette is correctly set to match the background color.

Bold size

How bold is bold? Here you can decide how thick bold type should be.

Bold

Character spacing

The number of points between characters. Below is a text with extra spacing. You can also set the spacing of the current line in the *Edit* menu by holding down the *CTRL* key and pressing the left and right cursor keys.

Spacing

REFERENCE: LAYOUT MENU

Italics

Here you can decide how italicized the letters should be.

Italics

Line spacing

The number of pixels between text lines. When you write a line of text and press *RETURN*, the cursor will move down a distance equal to the size of the font, plus the line spacing (A "pixel" is the smallest possible picture element in the resolution of the background image).

Grid size

From the *Edit* menu you can activate an invisible grid to which the text lines will lock onto when being moved. Here you can decide the size of the grid.

Outline size

The width of the outline, also measured in pixels.

Outline

Remap

When you use brushes or colorfonts, the palette of the brush or font may not match the palette of the background picture. When this option is turned on, Scala MultiMedia examines the palettes, and remaps the colors in the brush or font to match the existing background picture palette. This option will only work properly if there are similar colors in the two palettes. If you load a brush that has

different shades of red, while the background picture palette only has levels of blue, there will be no need to remap the brush, and no effect either.

Shadow length

The number of pixels between the original text and its shadow.

Shadow

Underline air

This is the open space between the descenders and the underlining.

Underline pos

This is the number of pixels from the top of the text and the underline.

Underline size

How many points wide should the underline be?



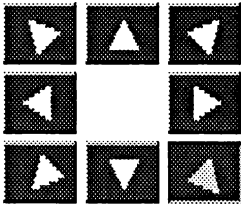
Here is an illustration that shows how the underlining parameters affect the text.

3D length

The depth of the 3D effect. The distance is measured in pixels in the direction of the projection (next page)

3D Length

REFERENCE: LAYOUT MENU



Shadow/3D Direction

To the left you can see the controls for shadow or 3D projection direction. Alternate between shadow and 3D by pressing the button below the direction buttons.



Here you see two different shadow directions, and the way they indicate where the imaginary light source is. Never use more than one shadow direction on the same page!

Margins

This is where you turn the margin settings on and off. When they are on, a tickmark is shown on the button. When turned off, the margins are ignored.

Word Wrap

Do you want Scala Multimedia to split the lines where necessary? When you write or import text, the cursor will automatically jump down one line when the cursor goes past the right margin.

Show boxes

When this option is turned on, you will see the empty text lines as boxes, and this can make it easier to move to the right line when filling text into a predefined layout. The boxes can be turned on by selecting *Show boxes* again, and they will not be visible when you run the script.

REFERENCE: LAYOUT MENU

Save Layout

You can save all the settings on the current picture by selecting *Save Layout*. Margins, text positions, colors, fonts, sizes etc. will then be saved. You can give this group of settings a name. By using *Load/Layout* from the *Edit* menu you can use the same layout in other pages. You will then have "empty" lines on the screen which can be filled with text, and each line will have the same colors, fonts and settings as the original line.

Save default

Saves the current layout as a "standard layout" that will be loaded every time you load a picture. To "forget" the standard layout, turn this option off again.

OK/Cancel

As usual *OK* confirms the changes, while *Cancel* allows you to return to the *Edit* menu without carrying out any of the changes.

About layouts

A "layout" is a feature in Scala MultiMedia that can be a real time-saver when creating presentations and video titles where lots of pages look similar. A layout contains information about the position of the text lines, typefaces, styles, justifications and colors, as well as margins etc. In other words, everything a normal page has, apart from the actual text.

When a layout has been loaded, you can use the up/down cursor keys to move between the empty lines. Use the *Show boxes* feature to see where the empty lines are.

REFERENCE: LAYOUT MENU

Combined with the *Load text* feature found in the *Load* menu, you can write your presentation in a word processor, and the load it into a Scala MultiMedia layout. If each page is supposed to have a headline, remember to include the **NEWPAGE** command in the text file to tell Scala MultiMedia to start writing on a new page.

A layout can be used in three different ways:

Saving a standard layout

In the *Layout* menu, there is an option called *Save standard*. When you turn this option on (the tickmark appears), the current layout information is saved to a file called "s:ScalaMM.layout". The next time you create a new page, Scala MultiMedia will load this layout and use it. When you don't want to use this layout any more, just turn *Save standard* off.

Saving a named layout

When you save a layout with a name, using the *Save Layout* command in the *Layout* menu, you can later retrieve it with the *Load Layout* command in the *Load* menu. You can use this command to build a library of layouts for different needs. We suggest that you save your layouts in the layout drawer, and also give them the suffix ".layout".

Connecting a layout to a picture

If you save a layout with a name that is identical to the name of the background picture, but with the addition ".layout", this layout will be loaded when you load the picture. The layout must be located in the same drawer as the picture, and *not* in the layout drawer.

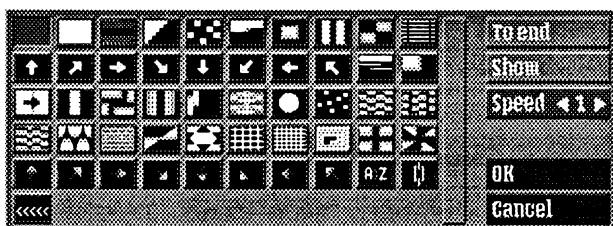
For example, if you save a picture with the name "Grass001.layout" in the "Backgrounds" drawer where the picture "Grass001" is located, this layout will be loaded each time you load the picture.

Text wipe menu

In this menu you decide how a line of text (or a brush) should appear or disappear on the page. Scala MultiMedia has a wide range of different wipes, ranging from simple "left-to-right" wipes, to the kind where the text moves in a specific direction and stops smoothly in the given position.

The wipe you select will only be applied to current line. That is, the line the cursor is situated on, or the selected line in the *List* menu. If you use the multiple selection feature in the *Edit* menu, the selected wipe will be applied to all text lines that are touched by the rubberband

You can reach the text wipe menu from the *IN* and *OUT* buttons on the left side of the *Edit* menu, or from the *Wipe* columns in the *List* menu. the Wipe in/out menus look the same, but will affect the text lines in a different way.



REFERENCE: TEXT WIPE MENU

For the sake of clarity we first explain the *Wipe* buttons in the *Edit* menu. Then we will take a look at how the in-wipes and the out-wipes work.

The default setting is that the text will be written on the picture before the page is displayed, and that it stays there until the page disappears. This is represented by the button in the top left corner of the *Wipe* menu. All the other buttons represent wipes that will take place after the page is displayed.

To select a wipe simply select the button representing it. The colors around the edge of the button will then change to simulate that the button is pressed in. The previously selected wipe button will pop out.

When you have selected the *Wipe in* menu from the *Edit* menu, the wipe type for other lines of text can be set by using the up/down arrow keys on the keyboard. The cursor will then move to the previous or the following line (the order of the lines can be edited using the *List* menu).

Show

To check if the selected wipe works the way you want it to, press the *Show* button.

To End

If you wish all the following lines of text on this page to use the same kind of transition you click on *To End*.

Speed

The speed of the transition can be regulated by selecting *Speed*. Set the speed by using the arrows; 1 is the slowest, and the maximum speed is between 10 and 50, depending on which wipe you have selected.

REFERENCE: TEXT WIPE MENU

Wipe in

The *Wipe in* menu is displayed when you select the *IN* button in the *Edit* menu, or a *Wipe in* button in the *List* menu.

Wipe out

The *Wipe out* menu is displayed when you select the *OUT* button in the *Edit* menu, or a *Wipe out* button in the *List* menu. If you select a wipe out in the *List* menu, a new line will be added to the bottom of the list. This line will have the same name as the original one, but it is used only to mark the out-wipe event. When you have selected a wipe, it will appear in the *Out* column on the selected text line.

Some special cases

Three of the wipes needs some further explanation, because they work in a special way.



A-Z If you for some purpose want to have a different wipe for each line of text, select the A-Z button. Scala MultiMedia will select the next available wipe on the lines which has the A-Z wipe. Try this wipe if you want to see *all* the available wipes.



Link This is not really a wipe, but rather a way to make several text lines or brushes appear at the same time. Apply this wipe to the lines following the line you want them to appear together with.



Crawl The text crawl is a ver powerful function that lets you scroll a text line of nearly unlimited length across the screen. The text will move from left to right.

When you type the text, turn on the crawl wipe, because then the text cursor will never reach the right side of the screen. The text will scroll to the left as you type.

If the cursor is located on the crawl line, and you select *Load Text*, the loaded text will be placed on the line.

REFERENCE: TEXT WIPE MENU

The fi	The first
The first line	The first line and the
t line	The first line and the second
and the	st line and the second
second	second

In the column to the far left, you see what happens if you in the Ei menu sets the in- and out-wipe f each line as it is written. The other column, we hav added the out-wipes after both lines we written.

REFERENCE: TEXT WIPE MENU

How to get the order right

As mentioned, Scala MultiMedia lets you control how text and brushes shall enter and leave the screen. The order of appearance is the same as the order they were written in. This is shown in the *List* menu, and to rearrange the order they appear, simply move the lines up or down in the *List* menu.

When you select a wipe out for a line from the *Edit* menu, the line will wipe out *after* the last line has wiped in. This means that if you start off with a blank page, write a line, and then select wipe in and out, this line will enter the screen, and then wipe out directly. If you enter another line in the same way, it will appear after the first line has disappeared, and then wipe out.

The other case would be if you first wrote the two lines, and then selected wipe in/out for both lines afterwards. Then both lines will first wipe in, and then both lines will wipe out.

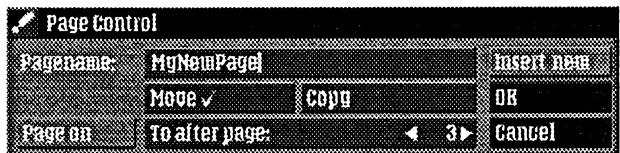
The easiest way to make things happen in the order you want them to, may be to use the *List* menu, since it shows you the text in- and out-wipes in a chronological order. You can easily rearrange the lines by just dragging them up or down in the list. See the *List* menu for more information about how to do this.

REFERENCE: TEXT WIPE MENU

The Page Control menu

This menu is used to move or copy the current page in the main menu. It also lets you turn off and on pages in the script if you temporarily don't want them to be displayed, and you can insert a blank line in the script (a *Direct Access Line*).

To get to this menu, select the *Page number* button in the *Main* menu for the page you want to copy or move.



Page name

You can easily change the name of the page by typing it directly into this field. This is particularly useful for pages that contains no pictures or animations. By default, Scala MultiMedia will name the page "Unnamed.n", but you can easily change the name in this menu.

REFERENCE: PAGE CONTROL MENU

Move

When you want to move a page up or down in the script, the easiest way is to just drag it up or down to the desired position. But with this technique you can't move the page outside the visible part of the script. If the script is long, you will have to move the page in several steps, which can take a long time.

To move a page longer, find first the number of the page you want to insert it after. Then, select the page number button of the page you want to move. The *Page control* menu appears. Select *Move*, and then set the *To after page* value to the pagenumber you wanted to move the page after. Click *OK*, and the page is moved.

Copy

This is an easy way to make one or more copies of a page in the script. Select *Copy*, and set the number of copies you want to make. Then use the *To after page* button to decide where the new pages will be inserted.

To after page

This button is used to decide where in the script a moved or copied page will be inserted.

Page on/off

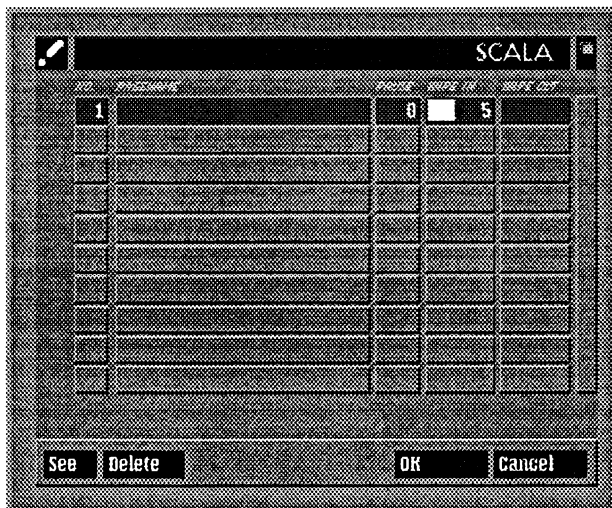
If you for some reason want Scala MultiMedia to *not* display certain pages when you run the script, use this button to turn the page off or on.

Insert new

This button inserts a *Direct Access Line* after the current line. Use this function to insert events in the middle of the script. This is useful if the *Direct Access Line* is outside the visible part of the script.

The List menu

This menu appears when you select *List* from the Text Menu. By using this menu, you can view the text lines in a similar way as the pages in the *Main* menu



You can easily set in/out wipes, pauses and rearrange the order of the text lines by clicking at the transition and pause buttons to the right.

REFERENCE: LIST MENU

When you are selecting a wipe in or wipe out, the *Text Wipe* menu is displayed. When this menu is displayed, you can select wipes for other lines in the script by just selecting the wipe button for the desired line. The *Text Wipe* menu will be updated to reflect the current wipe settings of the selected line. The same principle can be used in the *Wait* column.

To check the result, you can select Show from the *List* menu. Exit the *List* menu by clicking OK.

Line number

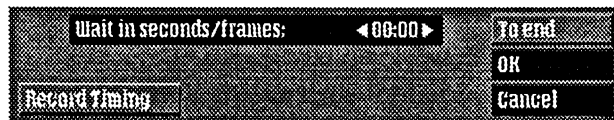
This is the number of the text line.

Text

This is the contents of the text line. If the line represent a brush loaded by the *Load Brush* command, the name of the brush is shown. Drag this button up or down to rearrange the order that the text and brushes appear in.

Pause

This is the amount of time since the previous text line started to wipe in. It works in the same way as the *Wait* buttons on the *Main* menu. You can set the pauses manually with the value button, or use the *Record timing* feature.



This can be very useful if you want to synchronize the wipe ins and outs to music or video. The page will be displayed, and you can click the right mouse button to make the text lines appear. The amount of time between your mouse clicks are saved in the *Pause* column. Press the *Escape* key to abort the process.

REFERENCE: LIST MENU

Wipe in

Select a button in this column to decide in which way the text line is to appear. The *Text wipe* menu will appear.

Wipe out

When you click a button in this column, the selected line in the script will be copied to the bottom of the list, and the *Wipe out* menu appears. Select how the line is to leave the page. If you want it to wipe out earlier, simply use the *Text* button and drag it upwards in the list.

Show

Select *Show* to view the page and the transitions.

OK/Cancel

Confirms or cancels your choices.

REFERENCE: LIST MENU

The Buttons menu

In this menu you can define interactive Scala Buttons on the screen. When you run the script, you can click these buttons, and SCALA will jump to the indicated picture and continue from that point of the script.

Button overview

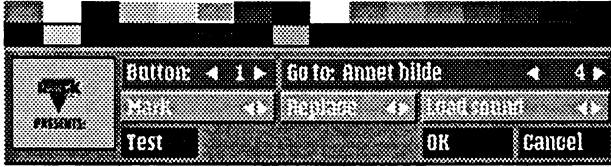
The Scala Button system looks simple, but is truly a very powerful system that makes you able to do almost anything. To utilize the full potential of the system, some technical and programming knowledge may be an advantage, but it is not required.

A button is no more than an interactive "hot spot". When the script is run, Scala MultiMedia senses where the pointer is, and if it is inside a button, the button can be highlighted to tell the user that it is possible to select this item. In this way, the user will always be aware of which choices he or she has. A special sound can also be added to the highlighting.

When the user clicks a mouse button, Scala MultiMedia will perform the actions you have decided for this button. It can be anything from a simple "goto page number", to variable checking/setting and conditional gotos. When you see how simple, yet advanced, the Scala Buttons are, you will soon be able to make very advanced interactive applications.

REFERENCE: BUTTONS MENU

Since Scala MultiMedia has built-in support for both Laserdisks, CD players, still video and MIDI, there is virtually no end to what you can do. Everything from simple quiz games and children training to huge interactive touch-screen applications is within your reach.



Button number

When you make buttons, they are given numbers from one upwards. The number on this button will increase by one for each new button you make. Use the increase/decrease arrows to select which button to work with. We call this button "the current button".

Go to

If you want Scala MultiMedia to jump to another picture in the script when the user selects a button, use the arrows on this button to select which page to go to. A miniature of the selected page is shown on the left side of the menu.

Mark/select

This button toggles between the options *Mark* and *Select*. As mentioned, there are two kinds of highlighting in Scala MultiMedia.

The first one, *Mark*, is the highlighting that is done when the pointer moves inside the square to indicate to the user that it is a button.

The other one, *Select*, is the highlighting that takes place when the user presses the mouse button. Then we can indicate that he or she has made a choice.

REFERENCE: BUTTONS MENU

There are two sets of highlight settings on this menu, and this button toggles between these two sets. This means that this button is *not* used to select if there is to be highlighting when the pointer is above the button *or* when the user selects it. It is simply used to toggle between the two views; *Mark* and *Select*. The settings of the *Highlight type* and *Set variable* buttons shows which highlighting to apply on either of the views.

Highlight type

This button toggles between the options Replace, Complement, Box, Fill and None. Use it to select which highlighting to use in the *Mark* and *Select* modes. The options are these:

Replace Use the color palette above the menu to decide which colors to replace the existing ones with. The top row shows the original colors, while the bottom row shows which colors to change to. First, click in the top row to select which color to change to. Then click in the bottom row, under the original color you want to change. The selected color is copied to the bottom row. You can for example use this feature to simulate 3-dimensional buttons that are pressed down, by changing the colors around the edge.

Complement This option turns the color palette "upside down" inside the button when it is highlighted. In a sixteen color palette, this means that color zero and color fifteen trades places, color one and color fourteen trades places, and so on.

REFERENCE: BUTTONS MENU

Set variable

If you want to set a variable when the user selects a button, the *Set variable* button will take you to the *Variables* menu. The selections you then do, will only be performed if the user selects this button when the script is run.

Show

Lets you test your buttons.

OK/Cancel

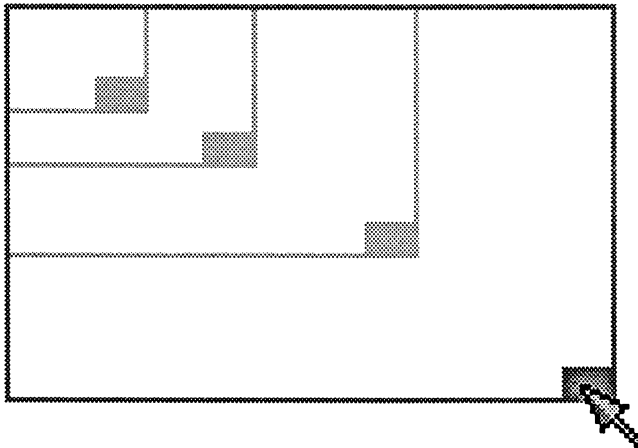
Confirms or cancels your choices.

How is it done?

Here is a step-by-step explanation on how to make and edit buttons

To make a button

When the *Buttons* menu is displayed, simply move the pointer to the desired location of the buttons upper left corner, and press the left mouse button. Keep it depressed and move the pointer down and left; Scala MultiMedia makes an outline square. This is a button! When the size is right, release the button. To make another button, repeat the process.



Click where you want the top left corner of the button to be, and drag the square down and right until the button has the desired size.

To move the button, click inside it and drag it to its new position.

REFERENCE: BUTTONS MENU

On the left side of the interactive menu you see the button's number. You can select another button by pressing the arrows on the *Button number* button.

On the right side of the button number there is a field which contains the name and number of the picture you want Scala MultiMedia to display when you select the indicated button.



Deleting buttons

To delete a button, simply press the DEL key. The current button will disappear.

Moving and resizing buttons

You can move and resize buttons easily using the mouse. To resize the button, select the square in the lower right corner of the button and drag it to the desired size. To move the button, point inside the button, hold down the left mouse button and drag it to the new location.

Mouse, joystick, keyboard and touch screen

Scala MultiMedia handles input from all these sources. In the *System* menu you can select *Joystick*, *Mouse*, *Function keys* and *Numeric keys* as input.

The *Mouse* option displays the pointer on the screen, and when the user moves the pointer inside a button, the *Mark* highlighting is displayed. When the mouse button is pressed, the *Select* highlighting is displayed.

When you use a joystick, one of the choices will always be highlighted (*Mark*), and when the joystick is moved, the next button in the indicated direction is marked. When the "fire button" is pressed, the *Select* highlight is displayed.

If *Function keys* option is turned on, you can also use the ten function keys at the top of the Amiga keyboard, named F1 to F10. These keys are "wired" to the ten first buttons - F1 will activate button number one, F2 will activate button number two, and so on.

REFERENCE: BUTTONS MENU

The numeric keypad on the keyboard can also be used. In this case, 1 activates button one, 2 activates button two, and so on. The keypad 0 activates button 10.

Touch screens come with software drivers that emulate the mouse. This means that you have to configure Scala MultiMedia to use the *Mouse* as input.

The Animation menu

What is Animation?

An animation is a series of pictures which are shown so quickly that they give the appearance of continuous movement. The Amiga is specially equipped to do just this, as it is very quick at dealing with graphics. There are a lot of programs for Amiga which make it very simple to make an animation. We will, however, not go into detail in this manual.

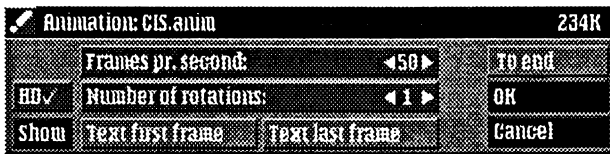
An animation is "delta compressed", which means that the pictures are not stored separately, but only the differences between the pictures are saved.

An animation can be everything from a simple, single-color rotating logo created in Deluxe Paint, to digitized video. The last kind requires more "power" to play back than the first one, because the number of different colors and pixels will be much larger.

To enable real-time playback of digital video, Scala AS has developed a proprietary 32-bit ANIM format that speeds up playback considerably. Converting animations to this format is done with the *AnimLab* utility, which is included in this package.

REFERENCE: ANIM MENU

Until now, animations have always been loaded into memory before playback. In Scala MultiMedia it is possible to play back virtually endless animations by loading them from disk as they play. This requires a fast computer and a fast hard disk, and may not in all cases work properly on an unaccelerated Amiga with a slow hard drive/controller card. Animations played from hard disk need to be indexed, in order to let Scala MultiMedia know how much it is necessary to load. Indexing of animations that is to be played directly from harddisk, is done with the *AnimLab* utility that comes with Scala Multimedia. If an indexed animation is loaded into Deluxe Paint or another animation program, and then saved, you will have to re-index the animation.



Number of Frames per Second

This is where you can select how many frames are to be shown per second. A normal speed is usually between 15 and 50. Motion at lower speeds than this may appear "jerky". The maximum speed is 50 on PAL systems and 60 on NTSC systems.

Number of Rotations

Animations are often made with the same beginning and end. Imagine, for instance, an animation being made up of a logotype rotating once around an x-axis. If this is repeated three times, the logotype will rotate three times (Not all animations are suitable for running more than once). You can choose how many times this should be repeated here.

Text First/Last frame

You can have text before or after the animation, in other words on the first or last frame of the animation. When it comes to texting the first frame of the animation you

should, however, be aware that if you place the text in a place on the picture which changes when the animation is being run, the text will be distorted during this. But pleasing effects can be achieved in certain situations (see below). Otherwise, text can be added to the first or last frames in the same way as for a normal background picture.

HD

Scala MultiMedia can play animations directly from a hard drive. This requires an accelerated Amiga (60020 or higher) to work well, and you also need a fast hard disk. Turn this option *ON* if you want to play the animation directly from the hard disk.

An animation that is to be played from the hard disk must be *Indexed* first. This is done with the "AnimLab" utility in the "Scala/Utilities" drawer.

Hard disk playback will be slower than if the animation is loaded into memory first (the default). Hard disk playback is best suited for animations that are so large that it is impossible to load them into memory first.

An interesting side effect

If you put text on the first page in an animation, there are ways to preserve the text while the animation is playing. If you know how pictures and animations are built, this will be easy to understand. If you have no knowledge of binary mathematics, don't expect to understand it at once.

All graphics are built from "bit-planes", which can be thought of as square areas in the memory. If a picture has two colors, you need one bit plane. Two bit planes give four colors, three bitplanes gives eight colors, and so on. The formula is to multiply two by itself the number of bitplanes. If you have one bitplane, this gives two colors, while two bitplanes gives $2*2=4$ colors. Four bitplanes gives $2*2*2*2$ colors, which equals sixteen.

REFERENCE: ANIM MENU

Each pixel location in each bitplane can have one of two values, on or off (zero and one). The corresponding pixels in each bitplane make up a *binary* number which represents the number of the color. The color number points to a location in the color palette. For example, if the different bits that make up a pixel gives the number 0011, this means that the color number in this pixel is $0*1+0*2+1*4+1*8$, which is 12.

When an animation is compressed, each bitplane is treated separately. This means that if all the "moving" colors are located in bitplane one and two, a text written in bitplane two or three will *not* be destroyed. What will happen, is that the color will change to a color that is the sum of the color number of the anim and the color number of the text. You can see an example of this in the *ScalaTime* demo.

The Execute menu

The Execute menu is available from a column in the *Main* menu. It lets you execute DOS, Workbench and ARexx programs. This makes it possible to perform functions that are not available within Scala MultiMedia. The menu looks like this:



The top left button is used to select what kind of command or application to execute. The available options are *Workbench*, *DOS* and *ARexx*.

- | | |
|------------------|---|
| <i>Workbench</i> | Executes a program that has a Workbench icon. |
| <i>DOS</i> | Executes a program that is available from CLI or Shell. |
| <i>ARexx</i> | Executes an ARexx command or an ARexx script. |

These settings are used by Scala MultiMedia to correctly run the selected program or command.

REFERENCE: EXECUTE MENU

Command

This field contains the name of the program you want to run. You can use the *Browse* button to find the program using the *File* menu, or you can type the name directly in into the text field.

Optional arguments will anyway have to be typed in.

Browse

This button lets you select a command using the *File* menu.

Wait

If you want Scala MultiMedia to wait until the command is executed before proceeding, turn this option on.

Show

Use this command to see if the command or script is executed correctly.

OK/Cancel

Confirms or cancels your choices.

ARexx + Lingo "macros"

An interesting possibility is to mix scripts and ARexx programs. If you are going to make a script that has parts that will be difficult to accomplish within Scala MultiMedia, these parts can be programmed in ARexx and run from this menu. These ARexx programs can use all Lingo commands, and will in principle work just like a normal script

An example can be if the script on some of the pages displays data from a database. To start making such a "macro", finish the screen layout of the pages, and save the script with the suffix ".rex". It will become an ARexx program, which can be run from the *Execute* menu.

Then you can modify this ARexx program to read data from the database, and display the information using the Lingo commands already in the script. See the *Lingo* section for more information.

The Variables menu

You reach this menu from the *Variables* column in the *Main* menu, or from the *Set variable* button in the *Button* menu. From this menu, you can create new variables, assign values using all ARexx operators and functions, and perform conditional jumps based on IF-statements. ARexx is required if you use this menu, and you have to run the "RexxMast" program before starting Scala MultiMedia

What is a variable?

The Scala definition of variables is closely resembled to ARexx, since all variable operators are performed by ARexx. Simplified, a variable is no more than a location in the computer memory where you can put a number or a text. The value is referred to via its name.

Variable names

The letter "x" is a typical variable name, but the name tells little or nothing about what it represents. Is x the number of cows in the field, is it the price of eggs, or is it the distance from the earth to the sun? It is easier to understand ("read") a program if the variable names are chosen with a bit more fantasy.

eggprice = 12

is much easier to understand than the line

x = 12

REFERENCE: VARIABLES MENU

(The reason for using single-letter variable names in our "button" tutorial, was because we had to type the variable name on the page in order to display the value, and since the value was between 0 and 9, a long variable name would complicate the text formatting - the lines would have been split).

Variable names are made from any of the letters A-Z and the numbers 0-9, plus underscore (`_`), dollar sign (`$`) and question mark (`?`). A variable name can not *start* with a digit.

Operators

All mathematical expressions that are allowed in ARexx can be used in Scala MultiMedia. If a variable contains only digits, you can use the arithmetic operators, like `+`, `-`, `*` and `/`. String variables can be concatenated with the `||` operator. There are many operators available, and you will find more information about them in your ARexx documentation.

Displaying variables

To display the contents of a variable, you can simply type the variable name on the screen using the *Edit* menu, with an exclamation mark (!) in front of it. When the page is displayed, the contents of the variable will be displayed, using the same typeface, justification and colors as the variable name was written with. The variable does not have to be on a line by itself, but can be embedded in a sentence, like:

My name is !name.

When the script is run, and the variable contains the word "Scala", the result will be:

My name is Scala

REFERENCE: VARIABLES MENU



able

button is used together with the *Set* button to assign a value to a variable. To select an earlier defined variable, use left/right arrow buttons to cycle through the variables. If you want to define a new variable, use the left/right arrow buttons to find an "open" space. You will see the cursor on the button, and you can type in a new name.

In this field you enter the value you want to assign to the variable that is shown on the *Variable* button. You are not restricted to a single value or mathematical expression. All ARexx functions and operators can be used.

For example, you can increase the value of a numeric variable by one, by selecting the variable in the *Variable* button, and then typing the variable name and "+ 1" in the field.

In addition to the usual mathematical operators, a wide range of ARexx-specific functions can be used. See the ARexx documentation for more information. The ARexx 2.0 manual has a comprehensive ARexx reference, and you will also find useful information in the ARexx/Lingo chapter of this manual. See tutorial two and interactive demo scripts for further examples.

IF...

Here you can type a logical expression, and if expression turns out to be *TRUE*, the GOTO to the right will be executed. You can have several IF-expressions, and will be a separate GOTO for each IF-statement.

You can define new expression in the same way as defined new variables. Use the left/right arrows or button to select an open space, and type away!

GOTO

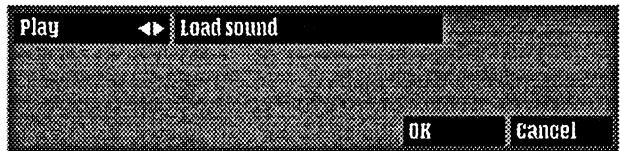
Use this button to set which page to jump to is the IF-true.

See the "ScalaQuiz" demo and study tutorial two to more about variables.

The Sound Menu

In the *Main* menu there is a column marked "Sound". Use the buttons in this column to control the sound events for every single picture in the script. You can load and start playing a sound, change the volume, or stop the sound. In addition you can make the script pause at the current picture until the sounds currently being played have stopped.

When you click a blank button in the *Sound* column, this menu appears



The upper left button in the menu is a "Command selector", and is used to select which sound command to apply. You leaf through the available options by using the left/right arrows. The options are: *Play, Volume, Stop, Wait* and *None*

REFERENCE: SOUND MENU

Load sound.

Use this option to load a new sound. Click the *Load Sound* button, and the file menu will appear, asking you to load a IFF sample, SMUS song or a DSS/Soundtracker module. Select a file in the file menu and click OK. A range of new buttons will then appear:



Play

The name of the selected sample/module will now appear in the *Load Sound* button, and the new buttons lets you control how the selected sound is to be played.

Volume button

In the middle you have the *Volume* button. This is a regular slider. When the slider is pushed all the way to the right, the sound volume is set to maximum.

Balance

The *Balance* button directs which stereo channel mono IFF samples will be played through: left, right or both. Soundtracker modules usually occupy all four channels, and can therefore not be moved left or right to specific channels. Stereo IFF samples will use one left and one right channel by default.

Fade in

If you want the volume to rise from silence to the level selected with *Volume*, use *Fade in* to select how many seconds it will take for the volume to reach this level. The default value is zero, which means that the sound will start to play directly, with no fade in period.

REFERENCE: SOUND MENU

Fade out

This button works in much the same way as the *Fade in* button, but adjusts the volume at the end of the sound instead of the beginning.

Pitch

To change the speed of IFF sample playback, use the *Period* button. The number is adjustable from 2000 (slowest) to 28000 (fastest). Scala MultiMedia will automatically set the pitch to the value that makes the sound play back at the "natural" speed. If the sound is sampled with a rate that is higher than allowed, the pitch will be set to 28000. During playback, the sound will be played too slow.

Loops

If you want the sample/module to be repeated more than once, use the *Loops* button. The default value is one, but you can adjust the number all the way up to 999 or infinite. The last setting will continue playing the sound until it is stopped with a *Stop* command.

Delay

It is possible to add a delay before the sound starts. For example, if you have an animation where there is an explosion after two seconds, set the delay to two. The default value is zero.

Show

To hear the sound and watch the picture that will be displayed along with it, select *Show*.

Wait

If you want the current picture or animation to be displayed until Scala MultiMedia has finished played this sound, click the *Wait* button so that the tickmark appears on the button.

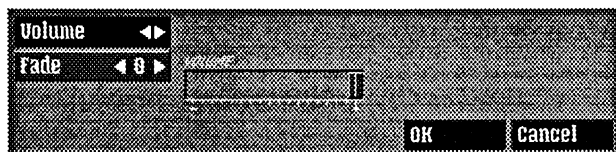
OK/Cancel

Confirm or cancel your choices.

REFERENCE: SOUND MENU

Volume

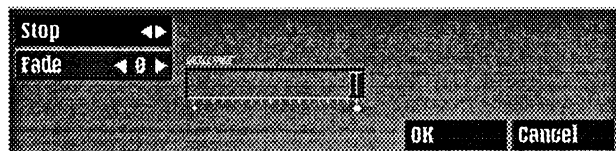
If you click the left arrow once in the *Play* menu, you will see the *Volume* menu.



Use this option to reduce or increase the volume of the sound that is currently playing. Use the volume slider in the middle of the menu to set the new volume. You can also set the length of the transition between the old and the new sound level shall take. Use the *Fade* button to set how many seconds you want it to take.

Stop

To turn off the sound, select *Stop*.

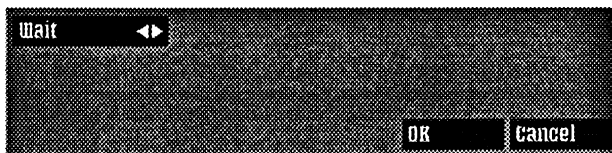


If you want the sound to fade out at the end, use the *Fade* button to set how many seconds it will take for volume to go from the current level and down to zero.

REFERENCE: SOUND MENU

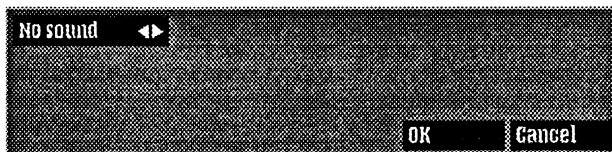
Wait

To display the current page in the script until the currently playing sound(s) are finished, select *Wait*.



None

If you want to remove a sound command, click the button in the *Sound* column in the *Main* menu. Then, use the left/right arrow on the command selector to select *None* and click OK.



A tips or two

Once the *Sound* menu is displayed, you can select sound commands for other pages in the script. Just click the sound button of the appropriate page in the script, and the settings in the sound menu will change to reflect the selected page. This speeds up your work, since you don't have to click *OK* for every sound command you enter. Just remember that selecting another line in the script while the sound menu is displayed, is the same as clicking *OK*. It is only the last sound settings you edit that will be reset when you select *Cancel*.

REFERENCE:SOUND MENU

Scala Lingo and ARexx

What happens when you create a script, save it to disk, and load it back again?

As you add pages, text, sound, interactive buttons etc., Scala keeps a description of the different elements in memory. This description covers everything from background pictures, color palettes, text, and font sizes to sound, interactive buttons, variables, and conditional branches -- and how the different parts relate to each other. When you save your script to disk, this information has to be stored in a clear and consistent manner. We have chosen to create our own language to describe scripts: Scala Lingo.

Most Scala users don't have to bother with Scala Lingo. You can create scripts, save them, run them, load them back and change them without even knowing that Scala Lingo exists. Every time you load a script, Scala reads the Lingo description in the file and builds the script in memory. Every time you save a script, Scala converts the script in memory to a Lingo description.

To get an idea of how Scala Lingo looks, try to look at a script file you have created. You can load it into your favorite text editor (such as CygnusED Pro from ASDG or the standard AmigaDOS ED editor), or you can view it from a "Shell" with the TYPE or MORE commands.

REFERENCE: AREXX AND LINGO

What's in it for me?

Some times, you may want to create advanced applications requiring special, tailor-made features, beyond Scala's built-in capabilities. Maybe you would like to link Scala with some other program, passing information back and forth. You could use Scala's powerful multimedia features to present information from database program, for example. Because Scala Lingo commands are available from ARExx, you can write ARExx macro programs controlling both Scala and other packages.

If you are happy working with Scala's friendly user interface, and don't need to expand Scala's functionality to fit any special needs, then you really don't need to bother with Scala Lingo.

Four different approaches to Scala Lingo

Let's take a look at the various ways to use Lingo:

Scala Scripts

As mentioned above, a Scala script file consists of a series of Lingo commands. As a user, you probably won't find it too exciting to create scripts by typing in Lingo commands, since this is much easier within Scala's elegant user interface. If you are a programmer, you may want to create programs with the ability to read, interpret or generate Scala scripts. In this case, you need to know Scala Lingo.

ARExx macros

These are ARExx programs that are executed at specific points of a Scala script, using the *Execute* menu. An ARExx macro may or may not talk to other programs (such as a database). During the execution of an ARExx macro, the Scala script halts and waits for commands from the ARExx macro. The macro may thereafter send Lingo commands back to Scala.

REFERENCE: AREXX AND LINGO

These Lingo commands may be as simple as setting a variable or making the script jump to a specific page, or as complex as generating new pages, adding text, sound, video or interactive buttons.

As an example of what ARExx macros can be used for, assume that you have made a big interactive application in Scala. In one part of the application you need to do a search in a database, and show the results on the screen. In this case, you would keep the entire application as a Scala script, and make an ARExx macro that searches in the database by sending ARExx commands to your database program.

If the result of the search is just a few lines of text, you can store the text in one or more Scala variables (using Lingo's SET command), and return to the Scala script. If the result of the search is more complex, (e.g. several pages of text, IFF pictures, sounds or laserdisk pictures), the ARExx macro can use Scala Lingo to show the pages, add text or sound, or show the video. Then, it would return control to the Scala script.

From other programs (via ARExx)

If you are mainly using some other program, but need multimedia functionality for some purpose, you can access those functions in Scala by sending Lingo commands to Scala's ARExx port. This requires that the program you are using has a "Send ARExx Command" feature.

Standalone ARExx programs. Entire applications can be written in ARExx, using Lingo to control Scala as a multimedia engine. This is not as hard as you might think, since Scala can generate ARExx programs from your scripts!

REFERENCE: AREXX AND LINGO

Scala Lingo and script files

A good way to start learning about Lingo is to examine some script files. All scripts start with the four characters "V2.0". This is a header that Scala needs to recognize the file as a script. Then a few global commands may follow; they describe settings which apply to the entire script.

There are global commands for turning the mouse pointer on or off, enabling or disabling interactive buttons, selecting input devices such as keyboard, mouse or joystick. Each of the global commands corresponds to a setting in Scala's System Menu.

After the global commands comes the body of the script: a series of "Events". An event describes what is often referred to as a page: a background picture or animation with text, sound, buttons, or other elements. Each event in the script file corresponds to one line in the *Main* menu.

Each event definition in the script starts with the keyword **EVENT**, followed by its name. The name is used as a reference to the event, so it has to be unique within the script. After the **EVENT** command come one or more commands describing the event's contents, followed by an **END** keyword which concludes the event definition. Here is an example of an event definition:

```
EVENT Background
PICTURE Work:Scala/Backgrounds/Fabrics001
PAUSE 3
END
```

This definition states that the event is called "Background", and consists of a background picture (Fabrics001). It will be shown for three seconds before the script proceeds with the next event. Most scripts contain more than one event. The events will be listed in the script file one after another, in the same order as they appear in the *Main* menu.

REFERENCE: AREXX AND LINGO

When you load the script back into Scala MultiMedia, all the commands are translated into an internal representation in memory. Any errors in the script (such as misspelled commands) will be reported. When the script is run, the events will be played back in the same order, starting over again from the top when the last event completes. There are some obvious exceptions to this: the script may contain interactive buttons, jumps or conditional branches which can affect the flow.

Scala Lingo and ARExx

Roughly all Lingo commands found in scripts can also be issued to Scala's ARExx port. There are also a few commands which are only available from ARExx. Scala's ARExx port is called `rex_x_ScalaMM`, that is, your ARExx programs must issue the command

```
address 'rex_x_ScalaMM'
```

before you can send commands to Scala. The port name must be typed in exactly as above, with upper case S and MM, and the rest in lower case.

To get started with Lingo and ARExx, load a script in Scala. Then save it again, but with a new name ending with ".rex" or ".arexx", for example "MyScript.ARExx". Scala will convert the script to a complete ready-to-run ARExx program! To run it, make sure Scala is started, open a Shell window in your workbench, and type "RX" followed by the name of the ARExx program. It will run and behave almost like a normal Scala script. Take a look at the program, and you will find that it looks very similar to the original Scala script. You will also notice that all commands are put within single quotes, like this:

```
'TEXT 80 110 "This is a line of text"'
```

This is to make sure that ARExx sends the line directly to Scala as a Lingo command, without trying to interpret the line as an ARExx command.

REFERENCE: AREXX AND LINGO

Even though the ARExx script looks very much the same and uses the same command language (Lingo) as the Scala script, there are some fundamental differences:

ARExx is an interpreted language, while Scala scripts are compiled to an internal format when you load the script. Scala scripts describe events in memory and their interconnections, while ARExx programs cause some immediate action to be taken each time a command is executed. Therefore, there is no EVENT...END command in ARExx. When using ARExx, there is just one event in memory. This event is nameless and is constructed dynamically.

Once an event has been played, it is deleted from memory, and a new event may be created.

The dynamic nature of ARExx gives more flexibility than normal Scala scripts, but can slow things down. It is also impossible to edit ARExx scripts in Scala's user interface!

Here is a smple ARExx script:

```
/* Show one page with text and sound */
address 'rexx_Scala@M'

'SOUND play Work:Sound/Samples/Applause'
'PICTURE Work:Scala/Backgrounds/Texture001'
'FONT FuturaC.font 78'
'COLOR 1 0 0 0 0 0 0'
'ATTRIBUTES shadow center'
'STYLE 0 3 4 3 6 2 1 65 8 2 20 0 5 0'
'TEXTWPE pers speed 5'
'TEXT 138 110 "Welcome to"
'TEXT 151 193 "Scala Lingo"
'TEXT 141 276 "and ARExx!"
'PAUSE 10'
'SHOW'
'SHOW OFF'
```

REFERENCE: AREXX AND LINGO

Notice that there are no EVENT and END commands. Instead, the SHOW command makes Scala show the event that has been created by the preceeding commands. The SHOW command returns after 10 seconds (because of the PAUSE 10 command), and then the script proceeds with SHOW OFF, causing Scala to leave playback mode.

A more advanced example of using Scala Lingo from ARexx can be found in the Scala/ARexx drawer. The file "Dir.scala" is an ARexx program which lists all files in a given drawer, using Scala. It demonstrates how to get information from another application (in this case AmigaDOS!), and display the result in Scala. Depending on how many file names there are, the program creates new pages and makes the layout dynamically. Note that you may have to change one line to make it run with your harddisk setup, see the comments in the file for further details.

To try the "Dir.scala" program, start it with "RX" from a Shell window. You may pass it one argument, stating which directory to scan, like this:

```
RX Work:Scala/ARexx/Dir.scala sys:
```

ARexx macros

Perhaps the most interesting use for Lingo and ARexx is running ARexx macros from a Scala script. Any ARexx program can be started as a macro from the *Execute* menu. Make sure the "Wait" option is selected if the macro contains Lingo commands. Try to start "Dir.scala" as a macro in one of your scripts. Once the macro starts, it will take over control from your script and show its pages. Press the right mouse button to proceed. After the last page has been shown, your script will proceed.

Some special Lingo commands have been made particularly for ARexx macros: SETVAR, GETVAR, GOTO and CONTINUE. These commands let your macro program communicate with the main script. See below for a complete description.

REFERENCE: AREXX AND LINGO

Using Scala Lingo from other programs

If you want to access Lingo commands from other programs (via ARexx), or from stand alone ARexx programs, all you have to do is to start Scala before starting the other programs. However, this also loads the user interface and brings up the *Main* menu, which takes time and memory. To save memory, you can start ScalaMMPlayer directly in a special ARexx mode, with the Shell command

```
run ScalaMMPlayer -arexx
```

The ScalaMMPlayer will hang around in memory waiting for ARexx commands. No screen or window will be opened. The only way to get rid of it is to send a QUIT command via ARexx.

This concludes the introduction to Lingo. Before covering each individual Lingo command in depth, we will take a look at Lingo's syntactic rules in general.

Scala Lingo syntax

A Scala Lingo command is one line of ASCII text, consisting of one keyword (the command itself), followed by a number of parameters. The keyword and parameters are separated by space characters. If a parameter includes space characters, it must be enclosed in double quotes ("). For example, the command

```
FONT GoudyB.font 48
```

consists of the keyword "FONT" and the two parameters "GoudyB.font" and "48". Since none of the parameters include spaces, quotes are optional. The command

```
TEXT 20 80 "Scala MultiMedia"
```

consists of the keyword "TEXT", "20", "80", and "Scala MultiMedia". Here, the quotes around "Scala MultiMedia" are required, because of the space.

REFERENCE: AREXX AND LINGO

Some characters have a special meaning in Lingo: the quote ("), the backslash (\), and the semicolon (;).

The quote is used to enclose string parameters as described above. If you need the quote character literally in a string parameter, you must type it in enclosed in backslashes: \"\ For example, if you want to display the text

Einar "Pitbull" Haugstad

at coordinates (100, 150) the command will be

TEXT 100 150 "Einar \"Pitbull\" Haugstad"

ARexx complicates the use of quotes a bit. When using Lingo from ARexx, the entire command line must be enclosed by single quotes ('). If your string parameter includes the single quote (apostrophe) character, you must double it in order not to confuse ARexx. For example, to display the text

Don't worry, be happy

the ARexx command will be

'TEXT 100 150 "Don''t worry, be happy"'

Scala will take care of all this when you save your script with a ".rex" or ".arexx" suffix, thereby generating an ARexx program.

The backslash has the following special meanings:

\"	inserts a quote character into a string,
\\t	inserts a tabulator character into a string,
\\	inserts a backslash character into a string,
	continues a string over to the next line.

REFERENCE: AREXX AND LINGO

The "continue string" operator is used to enter very long strings. For instance, a crawl text line may be of unlimited length, and is written

```
TEXT 0 200 "This is an extremely long line of text.\nIn fact, it is so long that we have to split it\ninto several lines in Lingo. The backslash character\nat the end of each line makes Lingo join the lines\ntogether, so they will appear as one long line\non the screen."
```

The semicolon is used for adding comments in the script. Everything from the semicolon and to the end of the line will be ignored by Lingo. If you need the semicolon literally in a string, make sure to use quotes.

Scala Lingo makes no distinction between upper and lower case letters. Keywords are written in upper case throughout this manual for the sake of clarity.

Scala Lingo commands

Below is an alphabetical list of Lingo commands. Each command is listed with its different parameters and options. The following conventions apply: Keywords are shown in upper case, and should be typed in exactly as spelled. Parameter values are shown in lower case, represent some value or string of your choice. Optional parts of the command are enclosed in [brackets].

Required alternatives are enclosed in {braces}, and separated by vertical bars (|).

REFERENCE: AREXX AND LINGO

Name: AACOLOR

Usage: AACOLOR *rgb*

Description: Define which color to antialias to.

Parameters:
rgb any RGB value (see the PALETTE command below for an explanation of RGB values).

Examples:

AACOLOR 000 (Antialias to black)

AACOLOR 777 (Antialias to gray)

See also:

ATTRIBUTES, PALETTE,
STYLE, TEXT

REFERENCE: AREXX AND LINGO

Name: **ANIM**

Usage: *ANIM name loops speed [DISK]*

Description: Controls playback of IFF animation files.

Parameters:

<i>name</i>	The name of the animation file.
<i>loop</i>	Number of rotations to play
<i>speed</i>	Playback speed in frames per second
<i>DISK</i>	Play the animations directly from hard disk, instead of loading the entire animation to memory before playing.

Examples:

```
ANIM "Work:Animations/Cartoon.anim" 1 12
ANIM "Video:DigitizedVideo" 1 25 DISK
```

See also:

WIPE

REFERENCE: AREXX AND LINGO

Name: ATTRIBUTES

Usage: ATTRIBUTES *attributes*

Description: Selects text attributes, such as Shadow ON/OFF, Italics ON/OFF etc. All the TEXT and BRUSH commands following an ATTRIBUTES command will get the attributes defined.

Parameters: Any selection of the following keywords can be used:

3D	The "3D" text style
ANTIALIAS	Antialias text
BOLD	Boldface
CENTER	Center text line
EDGE	The "Outline" text style
INACTIVE	The "Inactive" text effect
ITALICS	Italicized text
LEFT	Left justified text
NONE	No text attributes at all
REMAP	Remap brush or color font
RIGHT	Right justified text
SHADOW	The "Shadow" text style
UNDERLINE	Underlined text

Examples:

```
ATTRIBUTES SHADOW CENTER
ATTRIBUTES UNDERLINE EDGE SHADOW
ATTRIBUTES NONE
```

See also:

AACOLOR, COLOR, STYLE, TEXT

REFERENCE: AREXX AND LINGO

Name: **BLANK**

Usage: **BLANK** *width height depth screenmodes*
[backcolor]

Description: Defines a blank background page.

Parameters:

<i>width</i>	The width of the page in pixels
<i>height</i>	The height of the page in pixels
<i>depth</i>	Number of bitplanes

screenmodes:

Any selection of the following keywords:

LACE An interlaced screen

HIRES A hires screen

backcolor Which color to use as the
background color

Examples:

BLANK 640 512 4 hires lace 0
(16 color PAL hires interlace)

BLANK 320 200 5 3
(32 color NTSC lores, color 3)

See also:

PALETTE, WIPE

REFERENCE: AREXX AND LINGO

Name:	BRUSH
Usage:	BRUSH <i>x y name [PAUSE <i>p</i>] [LAST]</i>
Description:	Shows an IFF brush. The appearance of the brush will be affected by the preceeding ATTRIBUTES , COLOR , STYLE , and TEXTWIPE commands.
Parameters:	
<i>x y</i>	The coordinates where the brush should appear
<i>name</i>	The file name
<i>PAUSE p</i>	Delay or wait for mouse click before showing brush. The <i>p</i> parameter specifies number of seconds to wait; -1 means wait for mouse click.
<i>LAST</i>	The brush should appear on the last frame of an animation.

Examples:

```
BRUSH 80 100 "Work:Scala/Symbols/Lines/Line2"
```

```
BRUSH 80 100 "Work:Scala/Symbols/Arrows/Down1" PAUSE -1
```

```
BRUSH 80 100 "Work:Scala/Symbols/Arrows/Up2" PAUSE 3 LAST
```

See also:

ATTRIBUTES, **COLOR**, **STYLE**,
TEXTWIPE

REFERENCE: AREXX AND LINGO

Name:	BUTTON
Usage:	BUTTON [<i>POSITION</i> <i>x1 y1 x2 y2</i>] [<i>SET var expr</i>] [<i>action</i>]
Description:	Defines an interactive button, which the user can select with a mouse, touch screen, joystick or keyboard. The button will usually have a defined rectangle on the screen, which can be highlighted when the user clicks with the mouse inside the rectangle. The highlighting is defined in the preceeding MARK and SELECT commands, if any. When used in a Scala script, the BUTTON command can define certain actions to be performed when the user selects a button, such as setting a variable or jumping to a specific event. When used from ARexx, the BUTTON command does not define any actions. Instead, when the user selects a button, the SHOW command returns. The ARexx program can then test which button was selected by issuing a BUTTON command without parameters. Scala will return the number of button selected in the variable RESULT, or 0 if no button was pressed. This requires you to tell ARexx that you want results with the command OPTIONS RESULTS. Your ARexx program can test the value in RESULT and perform different actions depending on the value.
Parameters:	POSITION <i>x1 y1 x2 y2</i> Defines the size and position of the screen rectangle. (<i>x1</i> , <i>y1</i>) is the

REFERENCE: AREXX AND LINGO

coordinate of the upper left corner,
and (x2, y2) is the lower right corner.

Parameters not available from ARExx:

SET *var expr*

Set the variable 'var' to the value of the expression 'expr' when the user selects the button. See the SET command below for more information.

[*action*] can be one of the following:

GOTO *event*

Jump to a specific event when the user selects the button.

NEXT

Go to the next event when the user selects the button.

RETURN

Return to the previous event when the user selects the button.

Examples:

(In a Scala script:)

```
BUTTON POSITION 500 300 600 350 GOTO MainMenu  
BUTTON POSITION 100 90 170 120 SET score score+1  
GOTO ScoreBoard
```

(In an ARExx program:)

```
'BUTTON POSITION 500 300 600 350'  
'BUTTON POSITION 100 90 170 120'  
'PAUSE 30'  
'SHOW'  
'BUTTON'  
if result = 1 then MainMenu()  
if result = 2 then ScoreBoard()
```

See also: INTERACTIVE, JOYSTICK, MARK, MOUSE,

REFERENCE: AREXX AND LINGO

Name:	COLOR
Usage:	COLOR <i>text edge shadow 3d1 3d2 jam inactive</i>
Description:	Selects colors to be used for text foreground, outline, shadow, 3D, and inactive text. The parameters are numbers referring to colors in the palette, not RGB values. For example, 0 refers to the first color in the palette; 5 refers to the 6th color in the palette.
Parameters:	
<i>text</i>	The text color itself
<i>edge</i>	Outline color
<i>shadow</i>	Shadow color
<i>3d1, 3d2</i>	3D colors
<i>jam</i>	Not implemented.
<i>inactive</i>	Inactive color
Examples:	
	COLOR 1 0 0 0 0 0 0
	COLOR 1 5 5 11 12 0 4
See also:	ATTRIBUTES, STYLE, TEXT

REFERENCE: AREXX AND LINGO

Name:	CONTINUE
Usage:	CONTINUE
Description:	This command only works from ARexx macros, that is, ARexx programs started from a Scala script using EXECUTE. It causes the script playback to resume where it left off when the ARexx macro took over control.
See also:	GOTO

REFERENCE: AREXX AND LINGO

Name: CYCLE

Usage: CYCLE { *ON* | *OFF* }

Description: Enables or disables color cycling.

Parameters: *ON* or *OFF*

Examples: **CYCLE ON**
CYCLE OFF

REFERENCE: AREXX AND LINGO

Name:	END
Usage:	END
Description:	This command only works in Scala scripts. It terminates an event definition started with EVENT.
See also:	EVENT

REFERENCE: AREXX AND LINGO

Name: **EVENT**

Usage: **EVENT** *name* [*OFF*]

Description: This command only works in Scala scripts. It begins the definition of an event, which may include a series of Lingo commands. The definition is terminated with the **END** command.

Parameters:

name The name of the event. The event can be called almost anything, as long as the name is unique within the script.

OFF The event may be temporarily disabled. In this case, the event will be skipped when the script runs, and will be shown in another color in the main menu.

Example:

```
EVENT Test  
PICTURE Work:Scala/Backgrounds/Tech001  
PAUSE -1  
END
```

See also: **END**

REFERENCE: AREXX AND LINGO

Name:	EX
Usage:	EX <i>name command</i>
Description:	The Scala EX system lets external program modules extend Scala's functionality. To accomplish this, the EX modules must also be able to extend the Scala Lingo language. For example, the MIDI File Player lets you play back MIDI files from your Scala scripts. A command for playing back a MIDI file is required. This expandability is carried out by the "EX" command. It finds out which EX module is responsible for the command, and passes the entire command line over to the EX. Since the "EX" command is used to control any type of EX, you will have to check the individual EX's documentation to find out about the available commands and their syntax.
Parameters:	
<i>name</i>	The name of the EX, e.g. "MIDI", or "SonyLD"
<i>command</i>	The command to be passed over to the EX.
Examples:	<pre>EX MIDI PLAY Work:MIDI/TheRace.MIDI EX SonyLD AUDIO 1 EX PioneerLD PAUSE</pre>

REFERENCE: AREXX AND LINGO

Name:	EXECUTE										
Usage:	EXECUTE <i>command</i> [CLI] [WB] [AREXX][WAIT]										
Description:	Starts an external program. The program can be a CLI (Shell) program, a Workbench program, or an ARexx program. The ARexx program could be a macro which takes over control from the current Scala script. The program can be started in the background, or in the foreground (with the WAIT option). When you start a program or ARexx macro with the WAIT option, Scala will halt the current script, and enable normal mouse and keyboard input. This way, you can start programs that require user input. During execution with WAIT, Scala also accepts ARexx commands. After the command completes, the script resumes.										
Parameters:	<table><tr><td><i>command</i></td><td>The name of the program to start and any command line parameters</td></tr><tr><td><i>CLI</i></td><td>Starts the program with CLI startup</td></tr><tr><td><i>WB</i></td><td>Starts the program with Workbench startup</td></tr><tr><td><i>AREXX</i></td><td>Starts the program with ARexx startup</td></tr><tr><td><i>WAIT</i></td><td>Wait for command to complete before resuming script playback.</td></tr></table>	<i>command</i>	The name of the program to start and any command line parameters	<i>CLI</i>	Starts the program with CLI startup	<i>WB</i>	Starts the program with Workbench startup	<i>AREXX</i>	Starts the program with ARexx startup	<i>WAIT</i>	Wait for command to complete before resuming script playback.
<i>command</i>	The name of the program to start and any command line parameters										
<i>CLI</i>	Starts the program with CLI startup										
<i>WB</i>	Starts the program with Workbench startup										
<i>AREXX</i>	Starts the program with ARexx startup										
<i>WAIT</i>	Wait for command to complete before resuming script playback.										

Examples:

```
EXECUTE "sys:Utilities/Say Hello, this is Scala speaking!" CLI
EXECUTE "Work:Scala/ARexx Dir.Scala c:" AREXX WAIT
```

REFERENCE: AREXX AND LINGO

Name:	FKEYS
Usage:	FKEYS {ON OFF }
Description:	Enables or disables function key input for interactive buttons. With this feature enabled, the user can select interactive buttons by pressing the F-keys on the keyboard. F1 selects the first button, F2 the second and so on. Used in Scala scripts, this command is global, and affects the entire script.
Parameters:	ON or OFF
Examples:	FKEYS ON FKEYS OFF
See also:	BUTTON, INTERACTIVE, JOYSTICK, MOUSE, NUMKEYS

REFERENCE: AREXX AND LINGO

Name:	FONT
Usage:	FONT <i>name size</i>
Description:	Select a font for texting.
Parameters:	
<i>name</i>	The name of the font, e.g. "GoudyB.font"
<i>size</i>	The size of the font
Examples:	FONT FuturaL.font 85 FONT GoudyB.font 23
See also:	TEXT

REFERENCE: AREXX AND LINGO

Name:	GETVAR
Usage:	GETVAR <i>variable</i>
Description:	Gets the contents of a Scala variable and store it in an ARexx variable with the same name. If OPTIONS RESULTS is enabled, the contents will also be stored in the RESULT variable. This command can only be used from ARexx macros.
Parameters:	<i>variable</i> The name of the variable
Examples:	GETVAR score
See also:	SETVAR

REFERENCE: AREXX AND LINGO

Name: GOTO

Usage: GOTO *event*

Description: When used in Scala scripts, the GOTO command causes the script to jump to the specified event. When used in an ARexx macro, it causes playback of the current script to resume at the specified event. The GOTO command does not work from standalone ARexx scripts.

Parameters:
event The name of the event to jump to

Example:
GOTO Slideshow1

See also:
CONTINUE

REFERENCE: AREXX AND LINGO

Name: GRID

Usage: GRID { ON | OFF } *x_size* *y_size*

Description: This command only works in Scala scripts. The GRID command specifies whether the GRID function should be enabled when editing the page in Scala, and defines the grid size.

Parameters:

ON or OFF

x_size Horizontal grid size in pixels

y_size Vertical grid size in pixels

Examples:

GRID ON 60 50

REFERENCE: AREXX AND LINGO

Name:	IF
Usage:	IF <i>expression</i> [GOTO <i>event</i>] [RETURN]
Description:	This command only works in Scala scripts. If you need conditionals in your ARexx program, use ARexx' inbuilt IF..THEN command. The IF command is used to test variables and do jump somewhere else in the script depending on the result.
Parameters:	<i>expression</i> Any logical expression. Scala uses the same syntax as ARexx for expressions. <i>GOTO event</i> Causes the script to jump to the specified event. <i>RETURN</i> Causes the script to return to the previous event.
Examples:	<pre>IF x>3 GOTO "Game Over" IF "drink" = "Coke" RETURN</pre>
See also:	SET

REFERENCE: AREXX AND LINGO

Name:	INTERACTIVE
Usage:	INTERACTIVE { <i>ON</i> <i>OFF</i> }
Description:	Enable or disable interactive buttons in a script. Used in Scala scripts, this command is global, and affects the entire script.
Parameters:	<i>ON</i> or <i>OFF</i>
Examples:	INTERACTIVE ON INTERACTIVE OFF
See also:	BUTTON, FKEYS, JOYSTICK, MOUSE, NUMKEYS

REFERENCE: AREXX AND LINGO

Name: JOYSTICK

Usage: JOYSTICK { ON | OFF }

Description: Enable or disable joystick input for interactive buttons. With this feature enabled, the user can select interactive buttons by moving the joystick and pressing the button. Used in Scala scripts, this command is global, and affects the entire script.

Parameters:
ON or *OFF*

Examples:
JOYSTICK ON
JOYSTICK OFF

See also: BUTTON, FKEYS, INTERACTIVE, MARK,
MOUSE, NUMKEYS

REFERENCE: AREXX AND LINGO

Name: MARGINS

Usage: MARGINS { ON | OFF } *left_margin*
right_margin

Description: Set left and right margins. Serves two purposes: When showing left adjusted, right adjusted or centered text, the text will be adjusted according to the margins. The margins also affects editing in Scala.

Parameters:

ON or OFF

left_margin

Left margin position in pixels

right_margin

Right margin position in pixels

Examples:

MARGINS ON 20 620

MARGINS OFF

See also: ATTRIBUTES, WORDWRAP

REFERENCE: AREXX AND LINGO

Name: MARK

Usage: MARK *highlight*

Description: Defines how interactive buttons should be highlighted when the user moves the mouse or joystick. The MARK command affects the BUTTON commands following, until a new MARK command is found.

Parameters: One of the following highlight types:

BOX color Draws a box around the button. The color parameter specifies the color number to be used for drawing.

COMPLEMENT

Inverts all pixels within the button.

FILL color Fills the entire button with one color

NONE No visual highlighting

REPLACE colors

Replace some colors with others. This command requires a table of color numbers, one number for each color. For instance, a four-color picture requires four numbers. Color 0 will be replaced with the the first color in the list, color 1 will be replaced with the second, and so on.

Examples:

MARK BOX 1

MARK NONE

MARK REPLACE 0 8 2 3 4 5 6 7 1

See also:

BUTTON, MOUSE, JOYSTICK,
SELECT

REFERENCE: AREXX AND LINGO

Name:	MOUSE
Usage:	MOUSE { <i>ON</i> <i>OFF</i> }
Description:	Enables or disables mouse input for interactive buttons. With this feature enabled, the user can move the mouse pointer and select interactive buttons. If the buttons use a visual MARK highlight, the buttons will be highlighted as the user moves the mouse. Used in Scala scripts, this command is global, and affects the entire script.
Parameters:	<i>ON</i> or <i>OFF</i>
Examples:	MOUSE ON MOUSE OFF
See also:	BUTTON, FKEYS, INTERACTIVE, JOYSTICK, MARK, NUMKEYS

REFERENCE: AREXX AND LINGO

Name:	NUMKEYS
Usage:	NUMKEYS { <i>ON</i> <i>OFF</i> }
Description:	Enables or disables numeric keyboard input for interactive buttons. With this feature enabled, the user can select interactive buttons by pressing the numeric keys on the keyboard. '1' selects the first button, '2' the second and so on. If this feature is enabled, it is no longer possible to enter page numbers on the keyboard. Used in Scala scripts, this command is global, and affects the entire script.
Parameters:	<i>ON</i> or <i>OFF</i>
Examples:	NUMKEYS ON NUMKEYS OFF
See also:	BUTTON, FKEYS, INTERACTIVE, JOYSTICK, MOUSE

REFERENCE: AREXX AND LINGO

Name:	PALETTE
Usage:	PALETTE <i>rgb1 rgb2 ... rgbN</i>
Description:	Defines the color palette for a picture or animation. This command is required for blank pictures (defined with the BLANK command). It can also be used with IFF pictures or animations, to override the original color palette stored in the file.
Parameters:	<i>rgb1...rgbN</i> There should be one color value for each color in the picture. Each color is stored as a hexadecimal RGB value, that is, three hexadecimal digits specifying the Red, Green, and Blue component, respectively. For instance, 000 is black, 777 is middle gray, FFF is white, F00 is red, 00F is blue, F0F is purple, and so on.
Example:	PALETTE 000 fff ddd 97d 86c 75b 64a 539
See also:	BLANK

REFERENCE: AREXX AND LINGO

Name: PAUSE

Usage: PAUSE *seconds*

Description: Defines how long an event should be shown before going to the next. The time is specified in seconds, and is measured from the beginning of the event (that is, when the picture is wiped in). If all elements of the event are done after the specified amount of time, playback will move on to the next event. If the event is not finished, it will finish before proceeding. This means that all texts or loops of an animation are guaranteed to be complete before the next event is started.

Parameters: *seconds* Number of seconds to wait. This may be a fractional number, or -1. -1 will cause the event to pause until some external input (like mouse or keyboard input) causes the script to proceed.

Examples: PAUSE 3
PAUSE 1.5
PAUSE 0.268
PAUSE -1

REFERENCE: AREXX AND LINGO

Name:	PICTURE
Usage:	PICTURE <i>name</i>
Description:	Specify the name of an IFF picture file to be displayed.
Parameters:	
<i>name</i>	The name of the IFF file.
Example:	PICTURE "Work:Scala/Backgrounds/ Texture002"

REFERENCE: AREXX AND LINGO

Name: **POINTER**

Usage: **POINTER { *ON* | *OFF* }**

Description: Show or hide the mouse pointer.
Non-interactive applications will usually use **POINTER OFF**, but in some cases it may be useful to be able to move the mouse pointer around.

Used in Scala scripts, this command is global, and affects the entire script.

Parameters: *ON* or *OFF*

Examples: **POINTER ON**
POINTER OFF

REFERENCE: AREXX AND LINGO

Name: QUIT

Usage: QUIT

Description: This command is only available from AREXX, and is useful only in one case: When using ScalaMMPlayer directly from AREXX by starting it with "ScalaMMPlayer -arexx", the player will make itself resident and wait for AREXX commands without opening any screen or window. The QUIT command causes the ScalaMMPlayer to close down and exit, freeing all its memory.

REFERENCE: AREXX AND LINGO

Name: RETURN

Usage: RETURN

Description: Causes the script to return to the previous event. This command only works in Scala scripts.

See also: GOTO

REFERENCE: AREXX AND LINGO

Name:	SCRIPT
Usage:	SCRIPT <i>name</i>
Description:	Load and Execute a script file. The script will play once (one loop) and return.
Parameters:	
<i>name</i>	The name of the script file
Example:	SCRIPT "Work:Scala/Scripts/Slideshow.script"

REFERENCE: AREXX AND LINGO

Name:	SELECT
Usage:	SELECT <i>highlight</i>
Description:	Defines how interactive buttons should be highlighted when the user selects a button. The SELECT command affects successive BUTTON commands until a new SELECT command is found.
Parameters:	The SELECT command accepts the same parameters as the MARK command. See under MARK above for a list of parameters.
See also:	BUTTON, MARK

REFERENCE: AREXX AND LINGO

Name:	SET
Usage:	SET <i>variable expression</i>
Description:	Sets a Scala variable to the value of an expression. The expression can contain other Scala variables, which will be replaced by their value upon evaluation. When using SET from ARExx, the variable will be set during the SHOW command. Note that only the Scala variable will be set; no ARExx variables are affected. It may be more appropriate to use the SETVAR command from ARExx.
Parameters:	
<i>variable</i>	The name of the variable to set. Variable names must start with a letter (A-Z or a-z), and may contain letters, digits, or any of the characters "!", "?", "\$", and "_".
<i>expression</i>	The value to assign to the variable is obtained by evaluating an expression. This expression is a formula which may contain constant values, Scala variables, operators and functions. All operators and internal functions provided by ARExx can be used in expressions. The expression must adhere to ARExx' syntactic rules.

Examples:

```
SET x 0
SET score score+1
SET name "'John Smith'"
```

See also:

GETVAR, IF, SETVAR, TEXT

REFERENCE: AREXX AND LINGO

Name:	SETVAR
Usage:	SETVAR <i>variable value</i>
Description:	This command is only available from ARExx. It sets a Scala variable to some value. Unlike the SET command, SETVAR is performed immediately, and affects both the Scala variable and the ARExx variable with the same name. When Scala receives the command string from ARExx, no evaluation is performed on the value. Any calculations must therefore be carried out in your ARExx program before you send the command.
Parameters:	
<i>variable</i>	The name of the variable to set. The variable will be set both in Scala and in your ARExx environment.
<i>value</i>	The value to assign to the variable
Examples:	(In an ARExx program:) <code>'SETVAR score 0'</code> <code>'SETVAR name' result</code>
See also:	GETVAR, IF, SET, TEXT

REFERENCE: AREXX AND LINGO

Name: **SHOW**

Usage: **SHOW [OFF]**

Description: This command is only available from ARExx. It plays back an event which has previously been set up by a series of Lingo commands (such as PICTURE, TEXT, etc). The SHOW command returns after all elements of the event are completed, and the specified PAUSE has timed out. The user can abort playback by pressing a mouse button, hitting the special keys on the keyboard (such as Escape and the arrow keys), or by selecting an interactive button. In this case the SHOW command will return immediately. When SHOW returns, Scala will still be in playback mode, that is, any graphics will still be visible. You then have the choice of setting up and SHOWing a new event, returning to the Scala script with CONTINUE or GOTO (if your ARExx program was executed as a macro), or leaving playback mode with SHOW OFF. SHOW OFF closes down the screen and enables normal mouse and keyboard input.

Parameters: **OFF** Leaves playback mode

Examples: **SHOW**
SHOW OFF

REFERENCE: AREXX AND LINGO

Name: SOUND

Usage: SOUND [*command*][*settings*]

Description: Controls playback of sound files. The following file formats are supported: IFF 8SVX, IFF SMUS, GVP DSS, SoundTracker Songs, and SoundTracker Modules.

Parameters: A SOUND command consists of one command and an optional series of settings.

Commands:

PLAY <i>name</i>	Play a sound file
STOP	Stop all sounds currently playing
VOLUME <i>vol</i>	Set the volume of all sounds currently playing
WAIT	Wait for all sounds currently playing to complete

Settings:

DELAY <i>secs</i>	Delay before playing sound
FADEIN <i>secs</i>	Fade in sound
FADEOUT <i>secs</i>	Fade out sound
LOOPS <i>n</i>	Play sound <i>n</i> times
PAN <i>pan</i>	For mono sounds: set left, right or both channels playback
PERIOD <i>n</i>	Pitch
WAIT	Wait for sound playback to complete before proceeding with the next event

Examples: SOUND play "Work:Sounds/Step" volume 64 period 122
SOUND play "Work:Modules/MOD.Fresh" fadein 3 fadeout 10
SOUND stop fadeout 15
SOUND wait
SOUND volume 64

REFERENCE: AREXX AND LINGO

Name:	STYLE
Usage:	STYLE <i>dir shadowdir shadowlen 3Ddir 3Dlen outlinesize boldsize underpos undersize underair italics spacing linespacing</i>
Description:	Set various text style parameters.
Parameters:	
<i>dir</i>	Not implemented. Set to zero for future compatibility.
<i>shadowdir</i>	Direction of text shadow. Specified as a number between 0 and 7: 0 - North 1 - Northeast 2 - East 3 - Southeast 4 - South 5 - Southwest 6 - West 7 - Northwest
<i>shadowlen</i>	Length of the shadow
<i>3Ddir</i>	Direction of the 3D effect. Same format as shadow direction.
<i>3Dlen</i>	Length of the 3D effect.
<i>outlinesize</i>	Size of the outline effect.
<i>boldsize</i>	Size of the boldface effect.
<i>underpos</i>	Position of the underline
<i>undersize</i>	Size of the underline
<i>underair</i>	Underline 'air'
<i>italics</i>	Italics angle
<i>spacing</i>	Character spacing
<i>linespacing</i>	Line spacing

REFERENCE: AREXX AND LINGO

Style, continued

Examples:

STYLE 0 3 4 3 6 2 1 68 9 2 20 6 5 0

STYLE 0 3 2 3 6 2 1 19 3 1 20 0 5 0

See also:

ATTRIBUTES, COLOR, STYLE, TEXT

REFERENCE: AREXX AND LINGO

Name: TABS

Usage: TABS *pos1 pos2 ... posN*

Description: Sets the position of the Tabulator stops.
Up to ten tabs can be defined.

Parameters:

pos1 ... posN

A list of positions measured in pixels from the left. -1 disables a particular tab.

Examples:

TABS 50 100 150 200 250 300 350 400 450 500

TABS 100 450

See also: TEXT

REFERENCE: AREXX AND LINGO

Name:	TEXT
Usage:	TEXT <i>x y text [PAUSE pause] [LAST]</i>
Description:	Shows a line of text, with the current font and text attributes. The appearance of the text will be affected by the preceeding FONT, ATTRIBUTES, COLOR, STYLE, and TEXTWIPE commands. The text string may contain variable names, prefixed with an exclamation mark '!'. These will be replaced with the variable's value. For example, if the variable SCORE has the value 20, and the variable NAME has the value 'Daniel', the string "You scored !SCORE points, !NAME." will read: "You scored 20 points, Daniel."
Parameters:	
<i>x y</i>	The coordinates where the text should appear
<i>text</i>	The text string to show
<i>PAUSE pause</i>	Delay or wait for mouse click before showing text. The pause parameter specifies number of seconds to wait; -1 means wait for mouse click.
<i>LAST</i>	The text should appear on the last frame of an animation.
Examples:	
	TEXT 20 80 "Now is"
	TEXT 20 140 "the time" pause 1
See also:	AACOLOR, ATTRIBUTES, COLOR, FONT, SET, STYLE, TEXTOUT, TEXTWIPE

REFERENCE: AREXX AND LINGO

Name:	TEXTOUT
Usage:	TEXTOUT <i>number</i> [PAUSE <i>pause</i>] [LAST]
Description:	Removes a text line from the background. The preceeding TEXTWIPE command defines an out transition for the text.
Parameters:	
<i>number</i>	Which text line to remove. A text out transition must refer to a text line that was previously added with the TEXT command. For example, to remove the first text line, use the command TEXTOUT 1; to remove the fifth text line, use TEXTOUT 5 and so on. Note that texts appearing on the last frame of an animation (marked with the LAST keyword) have a separate count. For example, to remove the second line of text from the last animation frame, use the command TEXTOUT 2 LAST no matter how many texts were added to the first animation frame.

PAUSE *pause*

Delay or wait for mouse click before removing the text. The pause parameter specifies number of seconds to wait; -1 means wait for mouse click.

LAST The text to remove appeared on the last frame of an animation

Examples:

TEXTOUT 3

TEXTOUT 1 pause 2

TEXTOUT 2 pause -1 last

See also:

TEXT, TEXTWIPE

REFERENCE: AREXX AND LINGO

Name: **TEXTWIPE**

Usage: **TEXTWIPE** *name[options]*

Description: Define a transition for the following TEXTs, BRUSHes or TEXTOUTs.

Parameters:

name The name of the transition

options Most wipes can have several options, such as speed, direction etc. The wipe speed is set by including "SPEED n" in the command line, where n is a number between 1 and the maximum speed for the transition used.

See the "Scala Wipes" list below for a complete list of wipe names and options.

Examples:

```
TEXTWIPE cut  
TEXTWIPE diagonal speed 8  
TEXTWIPE bob west easein speed 4  
TEXTWIPE crawl speed 10 loops 3
```

See also: BRUSH, TEXT, TEXTOUT,
"Scala Wipes"

REFERENCE: AREXX AND LINGO

Name:	WIPE
Usage:	WIPE <i>name [options]</i>
Description:	Defines a transition for the picture or animation in the current event.
Parameters:	
<i>name</i>	The name of the transition
<i>options</i>	Most wipes can have several options, such as speed, direction etc. The wipe speed is set by including "SPEED n" in the command line, where n is a number between 1 and the maximum speed for the transition used. See the "Scala Wipes" list below for a complete list of wipe names and options.
Examples:	WIPE cut WIPE fade speed 10 WIPE push north damped speed 5
See also:	"Scala Wipes"

REFERENCE: AREXX AND LINGO

Name:	WORDWRAP
Usage:	WORDWRAP { <i>ON</i> <i>OFF</i> }
Description:	This command only works in Scala scripts. The WORDWRAP command specifies whether wordwrap should be enabled when editing the page in Scala.
Parameters:	<i>ON</i> or <i>OFF</i>
Examples:	WORDWRAP ON WORDWRAP OFF
See also:	MARGINS

Scala Wipes

On the next page is a list of all transitions available in Scala. Each wipe is listed with its name, maximum speed, and available options. An asterisk in the PICTURE column states that the wipe can be used with pictures and animations (WIPE command); an asterisk in the TEXT column states that the wipe can be used with texts and brushes (TEXTWIPE command). Some options can be combined, while other combinations are illegal. For example,

TEXTWIPE bob easein south

is legal, while

TEXTWIPE bob north south

is illegal.

REFERENCE: AREXX AND LINGO

NAME	PICTURE	TEXT	MAX SPEED	OPTIONS
backgammon	*	*	10	north south
blend	*		20	north south easein
bob		*	50	north northeast east southeast south southwest west northwest
brick	*	*	10	
center	*	*	10	
charmblend	*		20	north south easein
checker	*	*	10	north south
corner	*	*	10	northeast southeast southwest northwest
crawl		*	30	
cube	*		20	north south easein
curtain	*	*	10	
cut	*	*	1	
derreck	*		20	north south easein damped
diagonal	*	*	10	northeast southeast southwest northwest
diakon	*	*	10	east west
diamond	*	*	10	
dump		*	1	
escalator	*	*	10	
escalator2	*	*	10	
escalator3	*	*	10	
excrawl	*	*	10	north south
fade	*		10	
fadecut	*		10	
flipcoin	*		20	
flow	*		20	north south easein
funkis	*	*	10	
grid4	*	*	10	
grid8	*	*	10	
hugedots	*	*	1	north east south west
line	*		20	north south easein damped
link		*	1	
lowflipcoin	*		20	
newscrollin	*		20	north south easein damped
newscrollout	*		20	north south easein damped
next	*	*	10	
pers	*	*	20	
pokabal	*	*	10	

REFERENCE: AREXX AND LINGO

sh	*		20	north east south west easein damped
lon	*		20	north south easein
le	*		32	east west easein
allblend	*		20	north south easein
ooth	*		20	north east south west
ral	*	*	10	north east south west
lit	*	*	10	
ot	*	*	10	
angepush	*		30	north south easein
etch	*		20	north south easein
n	*	*	10	
perimpose	*		10	
leep	*	*	10	north south
iss	*	*	10	
	*		30	north south easein
n	*	*	10	north east south west
sdowns	*	*	10	east west
ndmill	*	*	10	
pe	*	*	10	east west
ord	*	*	10	

REFERENCE: AREXX AND LINGO

Scala EX overview



The Scala EX system is a revolutionary new way of adding new features to the software in an easy and flexible way. By utilizing this system, both Scala AS and third-party companies can develop controllers for new products and special users in a clean and quick way. There is no need to upgrade the entire Scala package - you only have to drop the icon of the EX into the right drawer on your hard disk. Then you are ready to go!

What can they do?

A Scala EX can do almost anything - both within Scala and in communication with external hardware and software.

Installation

In the Scala drawer on your hard disk there should be two drawers called "EX" and "Startup". The "EX" drawer is used to store the available EX modules. The ones you want to use, you simply move to the "Startup" drawer. These will be loaded by Scala, and available in the *Main* menu and in Scala Lingo.

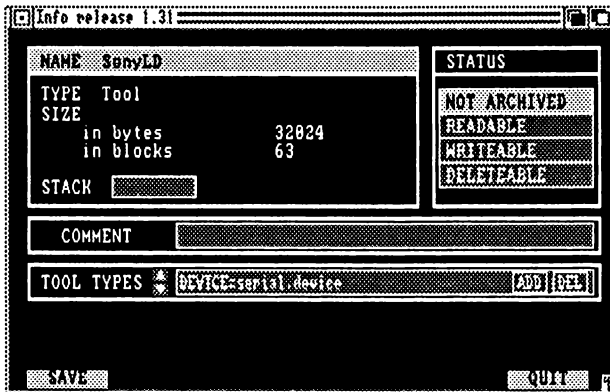
REFERENCE: SCALA EX

Configuration example

Some of the EXes may have to be configured to work properly with your setup. For example, if you use the MIDI EX to play music on a synthesizer, and have added a second serial port to communicate with a Sony Laserdisc player with the Sony LD EX, you will have to tell the EX to use the additional serial port. This is done by clicking once at Workbench icon, and selecting *Info* in the menu. For example, if you do this with the Sony LD EX, the *Tooltypes* field will contain the following lines:

```
DEVICE = serial.device
UNIT=0
BAUD=9600
```

See the documentation for the EX for more information on what each entry means. If you are using Workbench 1.3, the "Info" window will look like this:

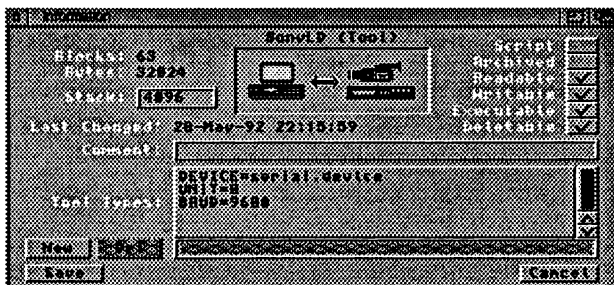


Only one of the lines is visible, so you have to use the up/down arrows on the left side of the *Tooltypes* text field to move up and down in the tooltypes list. For example, if you are going to change the unit number from zero to one, bring this line into view and click once with the left mouse button inside the text field. Then, move the cursor to the "0", press the *DEL* key on the keyboard to remove the number,

REFERENCE: SCALA EX

and type "1". Then, select *Save* to save the new settings.

If you use Workbench 2.0 or later, the Info window will look something like this...



To change a line, just click on it, and it will appear in the text field below. Use the keyboard to change the unit number from zero to one, and press the *RETURN* key on the keyboard. Then select *Save*.

Duplicating and renaming an EX

Let's say that you have two Sony Laserdisk players in your system. Then you will need two separate (but similar) EXes, and one of the Laserdisk players must be controlled through an additional serial port.

All you have to do is to duplicate the "original" EX, rename it, and change its settings as described above.

First, click the icon of the EX you want to copy, and select *Duplicate* or *Copy* in the pull-down menu. The Amiga will then make a copy of the EX, with the prefix "Copy of...". Change to a more appropriate name by selecting *Rename* in the pull-down menu, typing a new name, and pressing the *Return* key on the keyboard. The name you give the EX will appear in the Scala MultiMedia *Main* menu and in the script, so use a name that sounds logical.

REFERENCE: SCALA EX

Don't ever change the name of an EX without being aware of the consequences! If you do this by mistake, you may not be able to run a previously produced script, since there will be no EX with the name that is used within the script!



If an EX does not work properly, consult documentation of the EX and/or the peripheral that the EX is supposed to communicate with. Check to see if you are using the correct cables, and if the settings of the peripheral and the EX are correct.

Any good ideas?

If you have good ideas or wishes about things that a new EX could do, don't hesitate to tell us! You will find the name and address of Scala AS on the registration card in this manual.

SONY Laserdisc EX

Purpose

Controls the Sony LDP Series of VideoDisc players. You can control almost every function of the VideoDisc player: show a single frame, a sequence, select which audio channel to use, and turn the video picture on and off. This allows you to create interactive applications containing live video by mixing Amiga graphics and video using a genlock.

Required hardware

The following VideoDisc players are supported by the Sony LDP Controller:

- LDP-1500
- LDP-1500P
- LDP-1550
- LDP-1550P
- LDP-1600P
- LDP-1200
- LDP-3300P
- LDP-3600D

Connections & cables

The Videodisc player is connected to the Amiga's serial port with a standard RS-232 serial cable.

EX: SONY LD CONTROLLER

Control menu

Here is a description of the functionality of the menu, and each button. The top left button is the *Command selector*. You can use the arrows on this button to cycle through the available commands. The other buttons on the menu will disappear or appear, depending on whether or not they are needed.



Play

Use this command to play a sequence of video from the LaserDisk player. Use the control panel to wind back and forth to find the first and last frame you want to show, and then click the *In* or *Out* buttons to copy the value from the counter to the button.

If the *In* button is turned off (no tickmark), the playback will start from the frame it stopped on the last time the *Play* command was used.

If the *Out* button is turned off, the playback will not stop until a *Stop* or *Pause* command is issued.

If *Wait* is turned on, Scala MultiMedia will not proceed to the next page until the *Play* command has finished.

Stop

Halts video film playback and turns off the picture. Playback can be resumed with the *Play* command.

Pause

Halts video film playback and enters still mode. Playback can be resumed with the *Play* command.

Picture on/off

Turns the picture on or off.

EX: SONY LD CONTROLLER

Audio

Selects audio channel 1, 2, both or none.

Wait

Waits until video film playback has completed before proceeding to the next page.

Scala Lingo commands

EX SonyLD AUDIO [1] [2] [OFF]

Controls which audio channels should be active. "1" or "2" enables channel 1 or 2, respectively. Both channels can be enabled at the same time. "OFF" disables all audio.

EX SonyLD PAUSE

Halts video film playback and enters still mode. Playback can be resumed with the PLAY command.

EX SonyLD PICOFF

Turns the picture OFF.

EX SonyLD PICON

Turns the picture ON after the PICOFF command.

EX SonyLD PLAY [framenum1] [framenum2] [WAIT]

If both framenumbers are included, this command will play back a video film, starting from framenum1, ending at framenum2. If only framenum1 is included, the LaserDisk player will show this frame as a still picture. If no frame numbers are included, the player will start playback at the current position. This can be used to resume playback after a PAUSE or STOP command. The optional WAIT keyword causes the player to wait until the film is completed before proceeding with the next event.

EX: SONY LD CONTROLLER

EX SonyLD STOP

Halts video film playback and turns off the picture.
Playback can be resumed with the PLAY command.

EX SonyLD WAIT

Waits until video film playback has completed. If no video film is playing, the WAIT command will complete immediately.

Icon Tooltypes

DEVICE=serial.device

If you have a third party serial board in your Amiga, you may have to change this line. Consult your serial board manual.

UNIT=0

If you have a third party serial board in your Amiga, you may have to change this line.

BAUD=9600

The baud rate can be set to match your equipment. Consult your Sony manual on setting of DIP switches.

Pioneer Laserdisc Controller

Purpose

Controls the Pioneer VideoDisc players. You can control almost every function of the Laserdisc player: show a single frame, a sequence, select which audio channel to use, and turn the video picture on and off. This allows you to create interactive applications containing live video by mixing Amiga graphics and video using a genlock.

Required hardware

The following VideoDisc players are supported by the Pioneer Laserdisc Controller:

LD-V4100

LD-V4200

LD-V4300

LV-V8000

Connections & cables

The Videodisc player is connected to the Amiga's serial port with a standard RS-232 serial cable.

EX: PIONEER LD CONTROLLER

Control menu

Here is a description of the functionality of the menu, and each button. The top left button is the *Command selector*. You can use the arrows on this button to cycle through the available commands. The other buttons on the menu will disappear or appear, depending on whether or not they are needed.



Play

Use this command to play a sequence of video from the LaserDisk player. You can use the control panel to wind back and forth to find the first and last frame you want to show, and then click the *In* or *Out* buttons to copy the value from the counter to the button.

If the *In* button is turned off (no tickmark), the playback will start from the frame it stopped on the last time the *Play* command was used.

If the *Out* button is turned off, the playback will not stop until a *Stop* or *Pause* command is issued.

If *Wait* is turned on, Scala MultiMedia will not proceed to the next page before the *Play* command has finished.

Stop

Halts video film playback and turns off the picture. Playback can be resumed with the *Play* command.

Pause

Halts video film playback and enters still mode. Playback can be resumed with the *Play* command.

Picture on/off

Turns the picture on or off.

EX: PIONEER LD CONTROLLER

Audio

Selects audio channel 1, 2, both or none.

Wait

Waits until video film playback has completed before proceeding to the next page.

Scala Lingo commands

EX PioneerLD AUDIO [1] [2] [OFF]

Controls which audio channels should be active. "1" or "2" enables channel 1 or 2, respectively. Both channels can be enabled at the same time. "OFF" disables all audio.

EX PioneerLD PAUSE

Halts video film playback and enters still mode. Playback can be resumed with the PLAY command.

EX PioneerLD PICOFF

Turns the picture OFF.

EX PioneerLD PICON

Turns the picture ON after the PICOFF command.

EX PioneerLD PLAY [framenum1] [framenum2] [WAIT]

If both framenumbers are included, this command will play back a video film, starting from framenum1, ending at framenum2. If only framenum1 is included, the LaserDisk player will show this frame as a still picture. If no frame numbers are included, the player will start playback at the current position. This can be used to resume playback after a PAUSE or STOP command. The optional WAIT keyword causes the player to wait until the film is completed before proceeding with the next event.

EX: PIONEER LD CONTROLLER

EX PioneerLD STOP

Halts video film playback and turns off the picture.
Playback can be resumed with the PLAY command.

EX PioneerLD WAIT

Waits until video film playback has completed. If no video film is playing, the WAIT command will complete immediately.

Icon Tooltypes

DEVICE=serial.device

If you have a third party serial board in your Amiga, you may have to change this line. Consult your serial board manual.

UNIT=0

If you have a third party serial board in your Amiga, you may have to change this line.

BAUD=4800

The baud rate can be set to match your equipment. Consult your Sony manual on setting of DIP switches.

Canon ION Controller

Purpose

This EX controls the Canon ION StillVideo player (USA: Canon Xapshot). The player is connected to the Amiga's serial port via a special serial interface.

Required hardware

The following Canon still video players are supported:

- RV321
- RV311i
- RV311P
- RV311

You also need one of these serial interfaces:

- IA-V32
- IA-V31

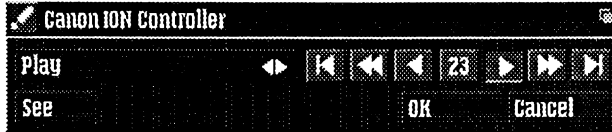
Connections & cables

The Amiga is connected to the Canon serial interface with a standard RS-232 serial cable.

EX: CANON ION CONTROLLER

Control menu

When you click a button in the *ION* column in the *Main* menu, the following menu appears:



The upper left button is the *Command selector*. Use the arrows on this button to select one of the following commands:

Play

Will show the picture indicated in the control panel to the right. You can use the control panel to select which picture to display.

Next

Will show the next picture on the disk.

Previous

Displays the previous picture on the disk.

Picture off

Turns the picture off.

Picture on

Turns the picture on. When you use the *Play*, *Next* and *Previous* commands, the picture is automatically turned on!

EX: CANON ION CONTROLLER

Scala Lingo commands

EX ION PLAY framenum

Shows a picture with a given number.

EX ION NEXTFRAME

Moves to the next picture on the disk.

EX ION PREVFRAME

Moves to the previous picture on the disk.

EX ION PICOFF

Turns the picture OFF.

EX ION PICON

Turns the picture ON after the PICOFF command.

Typical applications will only need the PLAY and PICOFF commands, since the PLAY command automatically turns on the picture.

Icon Tooltypes:

DEVICE=serial.device

If you have a third party serial board in your Amiga, you may have to change this line.

UNIT=0

If you have a third party serial board in your Amiga, you may have to change this line.

ADDRESS=FF

If you have more than one Still Video Player connected, you may have to change this line. Consult your Canon IA-V32 serial interface manual.

EX: CANON ION CONTROLLER

MIDI file player

Purpose

This EX lets you play high quality music and samplers on external MIDI synthesizers and samplers. The MIDI files can be made with a MIDI sequencer or composition program that supports this format (like Dr.T's KCS and Bard & Pipes from Blue Ribbon Soundworks).

The EX plays back MIDI File Format files of type 0, 1 and 2.

Required hardware

You will need a MIDI interface connected to the internal serial port, and of course, one or more MIDI compatible instruments with cables, amplifiers, speakers etc.

Connections & cables

The MIDI interface is connected to the internal serial port. Between the interface and your instrument(s) you use standard MIDI cables (DIN style). If you have several instruments and only one MIDI out connector, the instruments have to be daisy-chained.

See the manual of your MIDI equipment for more information.

EX: MIDI FILE PLAYER

The Control Menu

This is the MIDI player control menu.



Play

Will play a MIDI file. Click the *Load MIDI file* button to select which file to play. The file can be repeated several times by setting *Loops* to the desired number. The speed can be changed by adjusting the value in the *Tempo* button.

Wait

If *Wait* is turned on, Scala MultiMedia will wait until playback of the MIDI file has finished before proceeding to the next page.

Loops

The number of times to play the MIDI file.

Stop

Stops playback of the current MIDI song, if any.

Scala Lingo commands

EX MIDI PLAY filename

Plays back a MIDI file. Any other song currently playing will stop.

EX MIDI STOP

Stops playback of the current song, if any.

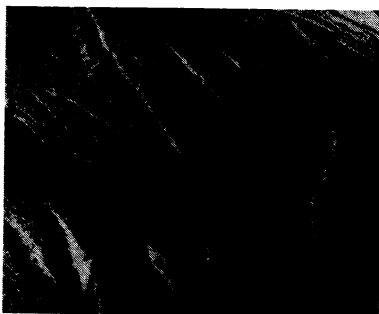
A P P E N D I X E S

- A Scala Art Library*
- B Scala Fonts*
- C Keyboard Shortcuts*
- D Utilities*

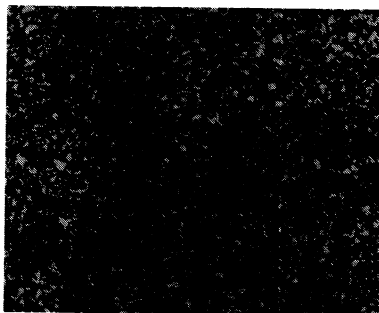
THE SCALA ART LIBRARY



abrics001

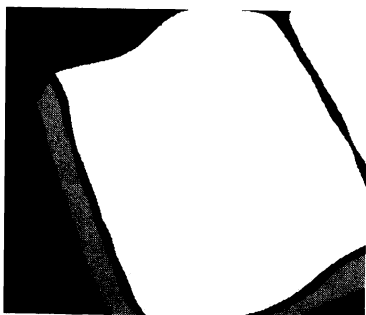


Fabrics002



abrics003

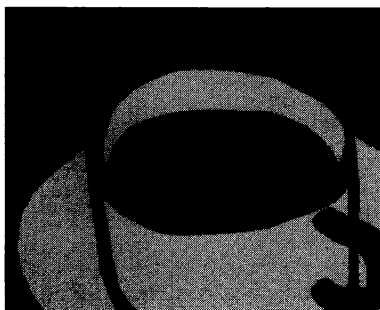
Fantasy001



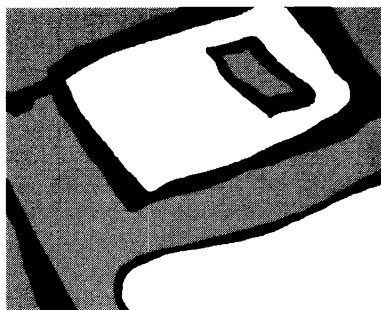
NaiveBook



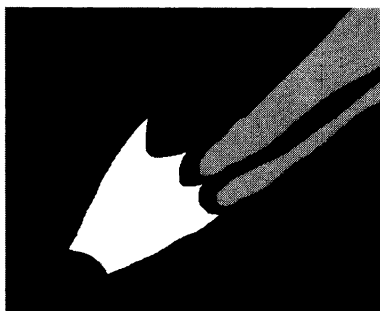
NaiveClock



NaiveCoffee



NaiveDisk



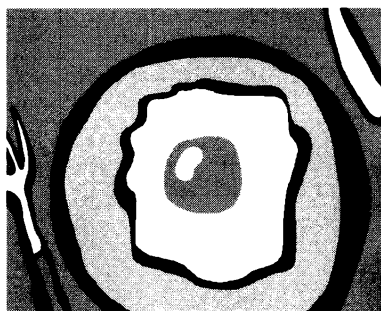
NaivePencil



NaivePhone



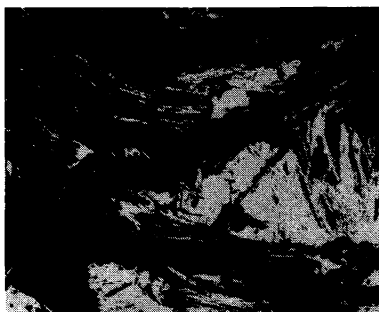
NaivePiano



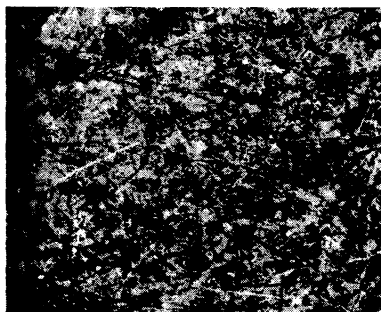
NaivePlate



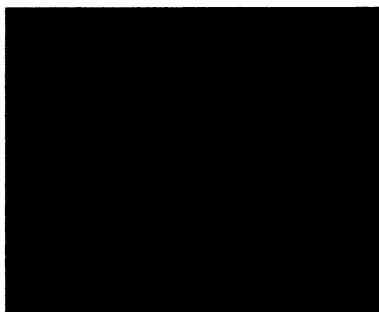
Painting001



Painting002



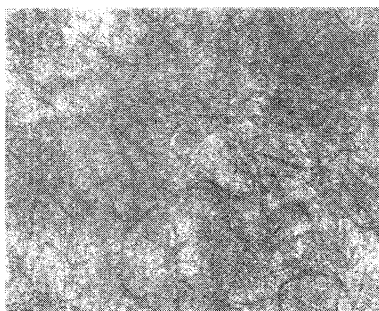
Painting003



Painting004



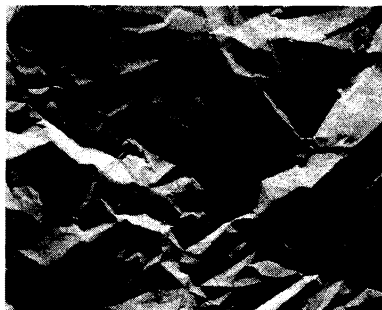
Paper001



Paper002



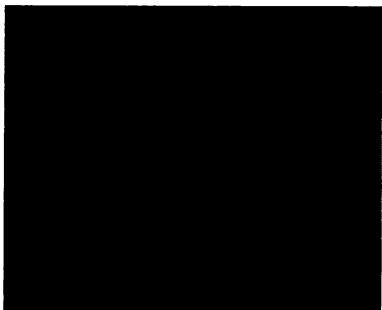
Paper003



Paper004



Paper005



Grass001



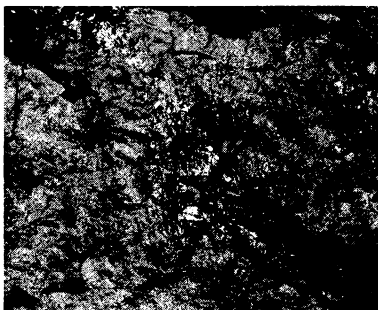
Sky001



Water001



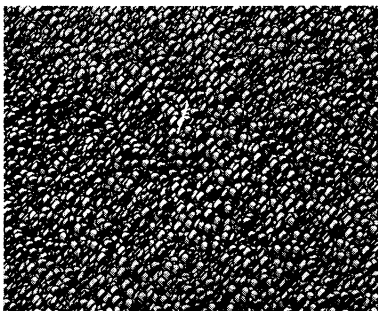
Stones001



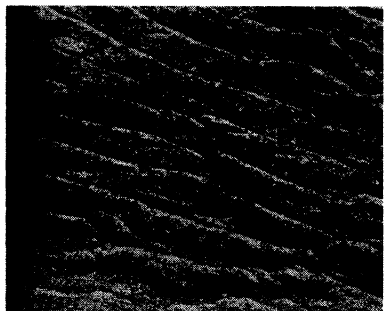
Stones002



Stones003



Stones004



Stones005



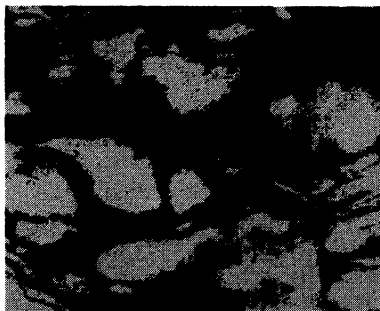
Stones006



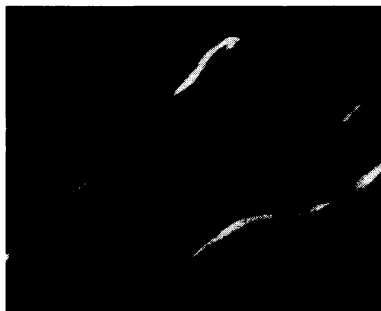
Stones007



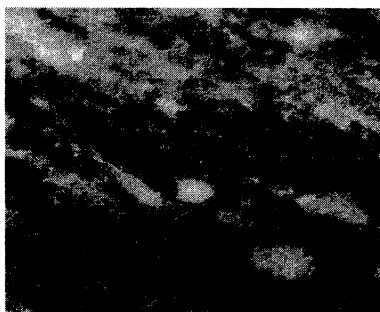
Stones008



Stones009



Stones010

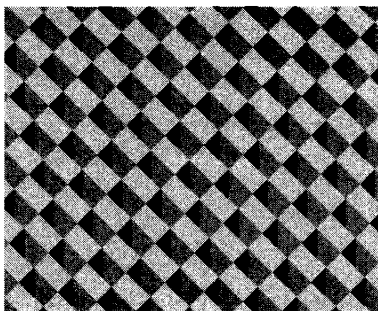


Stones011

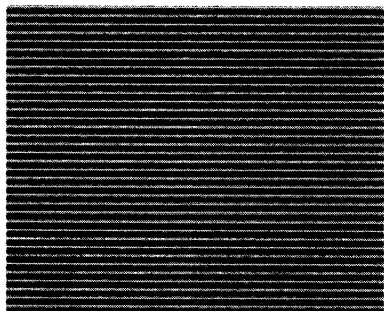
Tech001



Tech002



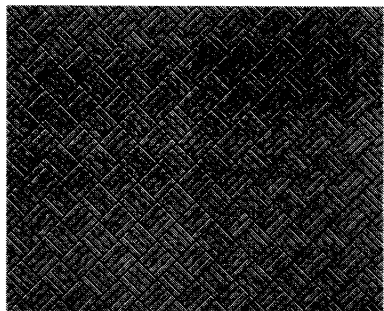
Tech003



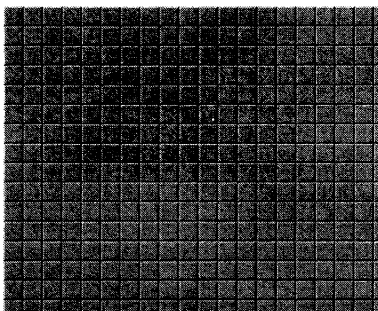
Tech004



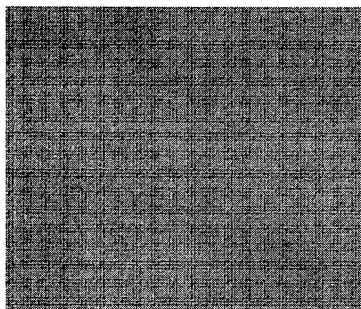
Tech005



Tech006



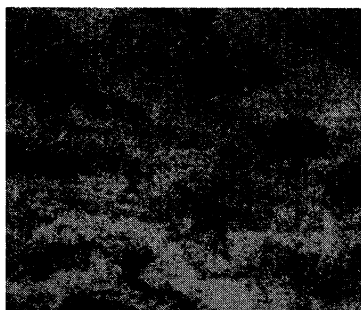
Tech007



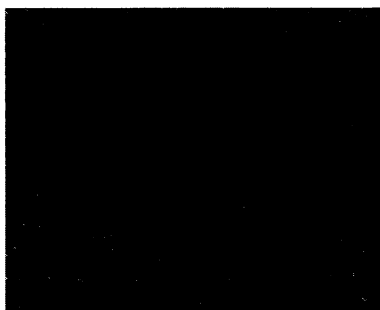
xture001



Texture002



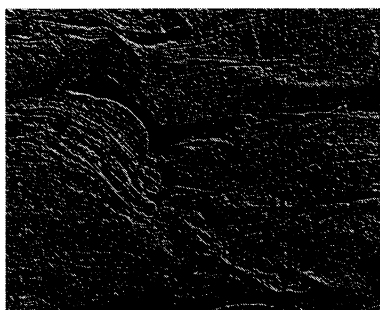
xture003



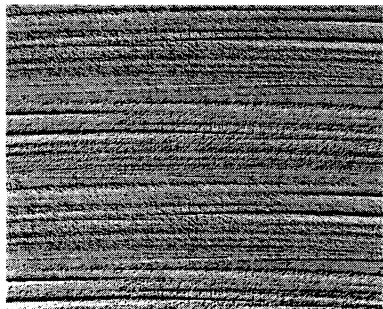
Texture004



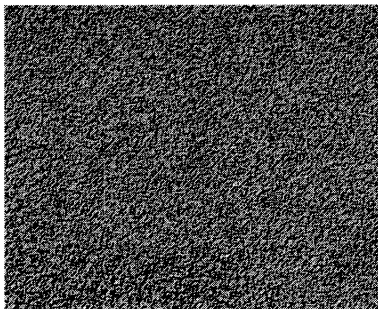
xture005



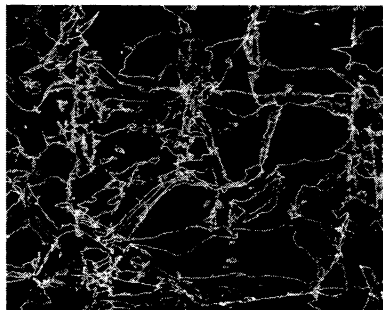
Texture006



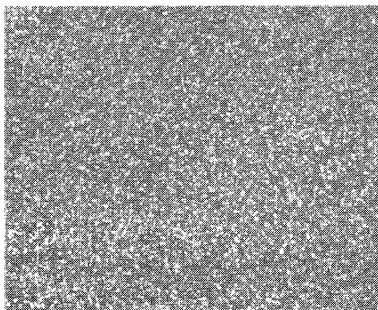
Texture007



Texture008



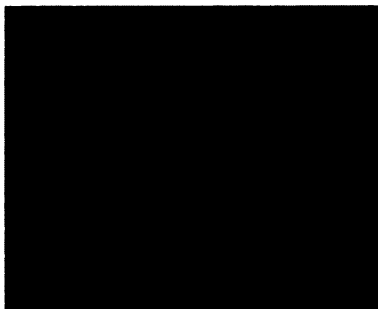
Texture009



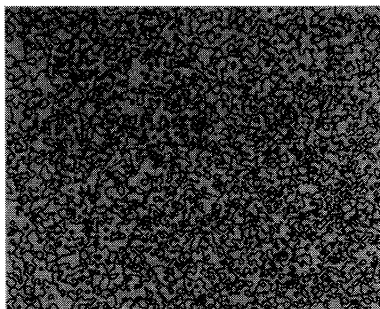
Texture010



Texture011

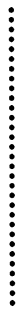


Texture012



Texture013

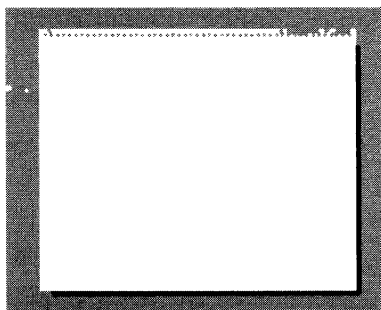
Texture014



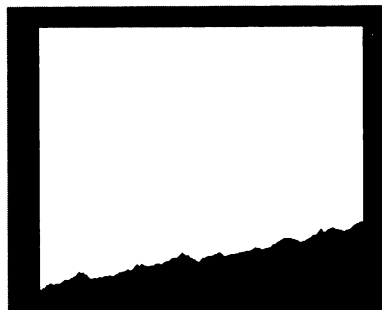
Theme001



Theme002



Theme003



Theme004

The Scala fonts

Fonts

The Scala fonts comes in a series of different sizes, and are designed to look good on computer screens and TV sets. On the following pages you will find the names and sizes of the fonts that are included in the Scala MultiMedia package. The fonts are shown in a resolution of 100 dots (points) per inch, and the size on the screen depends on the size and dot pitch of the screen.

More fonts

Scala A.S will release additional font libraries in the future. In addition, you can use all other regular Amiga fonts, and even colorfonts.

There is also full support for new "bullet" vector fonts included in Workbench 2.04, and in the *Font* menu you can scale these fonts to the exact point size you need. If you have a desk top publishing program that uses Agfa Compugraphic fonts, these fonts can be converted to the "bullet" format for use in Scala MultiMedia.

AaBbCcDdEeFfGgH

BetonC
44

AaBbCcDdEeFfGgHhIiJjKkLl
AaBbCcDdEeFf

Didot
28, 56

AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWw
AaBbCcDdEeFfGgHhIiJjKkLlMmNnO
AaBbCcDdEeFfGgHhIiJjKkLlM
AaBbCcDdEeFf

Compact
16, 24, 31, 64

AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz
AaBbCcDdEeFfGgHhIiJjKkLlMmNnO

CompactL
16, 31

AaBbCcDdEeFfGgHhIiJjKkLlMmNn
AaBbCcDdEeFfGgHhIiJjKkL
AaBbCcDdEeFfGgI
AaBbCcD

Franklin
18, 23, 36, 72

FranklinC
13, 28, 58, 114

AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz1234
AaBbCcDdEeFfGgHhIiJjKkLlMmNn
AaBbCcDdEeFfGgH
AaBbCcD

FuturaB
16, 20, 32

AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXx
AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQ
AaBbCcDdEeFfGgHhIiJjKkLlMm
AaBbCcDdEeFfGgHhI

FuturaC
22, 32, 78

AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPp
AaBbCcDdEeFfGgHhIiJjKl
AaBbCcDd
AaBbCcDd

FuturaL
20, 85

AaBbCcDdEeFfGgHhIiJjKlMmNnC
AaBbCcD

AaBbCcDdEeFfGgHhIiJjKkLl

AaBbCcDdEeFf

AaBbCc

FuturaX
24, 47, 94

AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPp

AaBbCcDdEeFfGgHhIiJjKkLlM

AaBbCcDdEeFf

Garamond
23, 32, 64

AaBbCcDdEeFfGgHhIiJjKkLlMmNnO

AaBbCcDdEeFfGgHhIiJjKkLl

AaBbCcDdEe

AaBbCc

Gill
16, 24, 57, 96

AaBbCcDdEeFf

GillN
58

GoudyB
23, 48

AaBbCcDdEeFfGgHhIiJjKkLl
AaBbCcDdEeFf

GoudyL
23, 48

AaBbCcDdEeFfGgHhIiJjKkLlMm
AaBbCcDdEeFf

HelveticaN
39, 78

AaBbCcDdEeFfGg
AaBbCcDdEeFf

NewsGothic
56, 114

AaBbCcDdEeFfGg
AaBbCcDdEeFf

S C A L A F O N T S

Keyboard shortcuts

Even if all commands can be executed from the menus, there are a number of keyboard shortcuts that can help the advanced users save time.

Function keys 1-10

The function keys are located across the top of the keyboard, and are labelled F1 to F10.

Amiga-key combinations

A number of commands can be executed by holding down the right *Amiga*-key and pressing another key. The right *Amiga*-key is located on the right side of the spacebar.

Other

In addition, there are a number of keys that have special functions, like ESCAPE, ENTER and SPACE.

On the next three pages you will find a complete list of all the keyboard shortcuts in Scala Multimedia.

KEYBOARD SHORTCUTS

IN THE EDIT MENU

A + y	Load layout
A + e	Load symbol
A + l	The load menu
A + r	Load background picture
A + k	Converts upper case letters to lower case and vice versa.
F3	Saves the page and makes a new page with the same background picture.
F4	Saves the page and makes a new page which is an exact copy.
F10	Redraws the text
A + DEL	Clears the page
A + -	Selects justification: free, left, center and right.
Alt + left	Moves the cursor one word left.
Alt + right	Moves the cursor one word right.
F1, F2	Go to previous page / next page.
Shift DEL	Deletes the line from cursor position to end of line.
Shift BACKSP	Deletes the line from the start of the line to the cursor position.
A + x	Deletes the line and puts it in the paste buffer.

KEYBOARD SHORTCUTS

A + c	Copies the line to the paste buffer.
A + p	Copies the paste buffer to the current line.
A + b	Bold on/off.
A + i	Italic on/off.
A + u	Underline on/off.
A + o	Outline on/off.
A + w	Shadow on/off.
A + 3	3D on/off.
A + v	Views the current picture and its text transitions.
A + s	The save menu
A + a	Copies attributes
A + g	Gives attributes to the current line.
A + m	Equivalent with the <i>Move</i> command in the text menu.
Ctrl+left arrow	Decreases the character spacing on the current line.
Ctrl+right arrow	Increases the character spacing on the current line.
Backslash	Manual kerning.
RETURN	Splits the line at the cursor position, and moves the rest of the line down.
ENTER	Splits the text line at the cursor position, without moving anything.

KEYBOARD SHORTCUTS

IN THE EDIT MENU

SPACE BAR	Run the script
F5	Insert a blank line after the selected one.

WHILE RUNNING A SCRIPT

ESCAPE	Stop the script and return to the <i>Main</i> menu.
SHIFT + ESCAPE	Stop the script and enter the <i>Edit</i> menu with the page that was displayed when you pressed the keys.
Number + RETURN	Jumps to the page with the typed number. For example, to go to page number 23, press 2, press 3, and then press the RETURN key.

Scala MultiMedia utilities

In the "Scala/Utilities" drawer, you will find some helpful utilities that will help you

ScalaMMPrint

This is a utility that can make a printout of your script, on a color or black&white printer. The scripts can be output as pictures or text, in many different formats. ScalaMMPrint also has built-in support for Postscript printers.

ScalaMMPrint is activated from the *System* menu, or from the icon in the "Scala/Utilities" drawer.

AnimLab

This program is capable of converting your animations to the 32-bit ANIM format that can be used by Scala MultiMedia. The converted file will play back smoother than the original 8-bit ANIM file.

AnimLab is capable of converting animations between PAL and NTSC formats, and has also other advanced features.

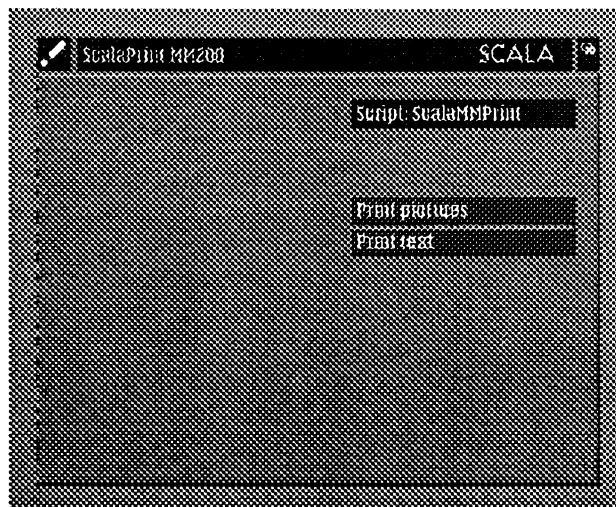
FixScript

If you have moved pictures used in a script to other drawers on your hard disk, Scala MultiMedia will not find them when you run the script. *FixScript* is a utility that searches your disks to find the missing pictures, sounds, brushes etc.

Utilities: ScalaMMPrint

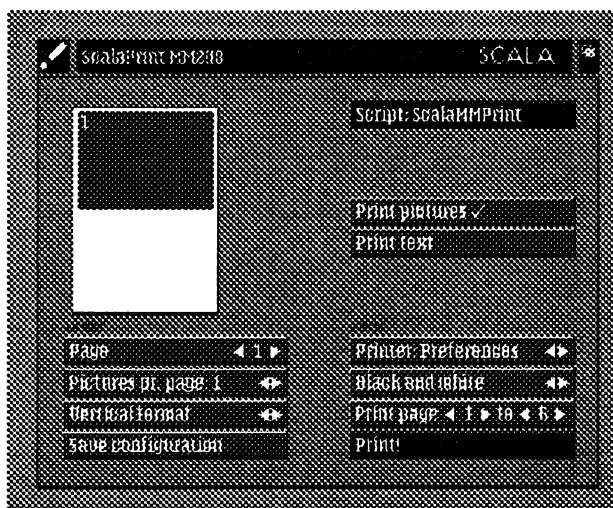
ScalaMMPrint

Scala MultiMedia system also includes ScalaMMPrint, an utility program for making hard copies of your scripts. ScalaMMPrint supports a wide range of output devices. In addition all Preferences printers, it also supports Postscript laser printers. ScalaMMPrint is activate from the System menu. This is the ScalaMMPrint main menu



ScalaMMPrint is by default loading the current script. You can select another one by pressing the button with the script name. Below this button, there are two buttons you use to select if you want to output the script as text or graphics. If you select *Print Pictures*, the screen on the next page will appear.

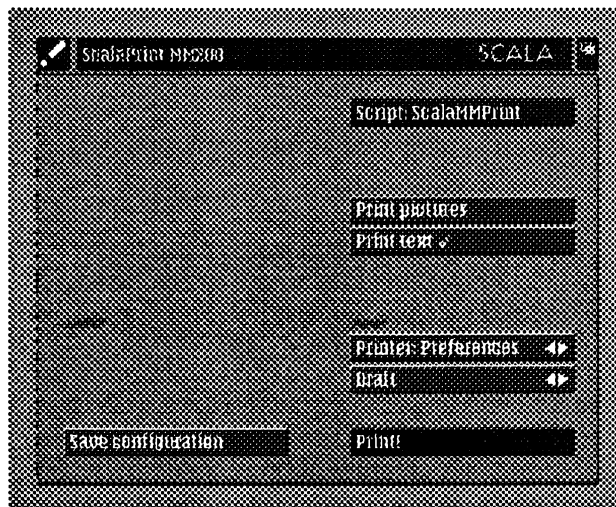
Utilities: ScalaMMPrint



On the top left side of the screen you see a representation of the output page, with shaded squares indicating the position of the pictures. In the lower half of the screen there are some buttons:

<i>Page</i>	Use the arrows on this button to select output page.
<i>Pictures pr. page</i>	Selects the number of pictures on the selected output page.
<i>Vertical format</i>	Toggles between vertical and horizontal format.
<i>Save configuration</i>	Saves the current settings in <code>s:scala.configuration</code>
<i>Printer</i>	Selects type of printer: PostScript to parallel or serial port, or the printer selected in Preferences.

Utilities: ScalaMMPrint



Grayscale This button toggles between color, greyscale and black & white. Greyscale is the only valid setting for PostScript printers.

Print page ... to ... Select the first and last page to be printed.

Print! Starts the print job. Remember to check that the printer is turned on and "On-line"!

If you selected to print the script as test, the menu above lets you decide the printing parameters:

The differences between this menu and the previous, is that you here select the font and font size for PostScript printers, or draft or NLQ quality for preferences printers.

Scala MultiMedia INDEX

)	113, 162, 180-181, 230, 235, 266
niga	2-4, 8-11, 16, 33, 67, 80, 103-104, 106-113, 116-117, 121-122, 125, 142, 144, 147, 149, 157, 170-172, 200-204, 218, 224, 280, 282, 285-286, 289-290, 292
NIM	4-5, 7-8, 13-15, 18, 20-21, 48, 57-60, 93, 103-109, 112-115, 119, 126-129, 134-137, 141, 150, 154, 173, 190, 202-205, 214, 221, 229, 232, 254-255, 269-274
mination menu	119, 136-137
ntialias level	178
ntialiasing	108, 173, 178
rex	6-7, 14, 114-115, 119, 206-210, 218-277
ld size	178
ush	6, 27, 158, 165, 179-180, 184-189, 193, 230, 232, 271, 274
ttions	6-7, 18, 20, 22-23, 26, 36, 67, 73, 77, 83-84, 90, 96-97, 101, 118-132, 148, 152, 154, 156, 161, 181, 184-185, 192-193, 196-201, 210-213, 218, 220-222, 242, 248-253, 261, 283, 287
non ION	6, 290-293
OTV	6, 116
aracter spacing	79-80, 178, 266
lor bar	29, 32, 43-44, 56, 85
lor cycling	109, 173, 237
lor palette	55-56, 119, 124, 153, 161, 163, 170-175, 178, 198, 205, 218, 254
lorfonts	170-171, 179
nfiguration	127, 157, 279
unter	73, 93, 100, 153, 283, 287
ursor	19, 24-29, 33, 50, 64, 78-82, 94, 124-125, 149, 159-161, 168, 178-179, 181-182, 184-186, 210, 279
CTV	109
irect Access Line	128, 134, 190-191
nble-click	3, 10, 12, 16, 19, 54, 63, 134-135, 137
awer	3, 10, 12-16, 37, 39-40, 55, 57, 69-71, 75, 117, 132, 140-142, 146-149, 155-156, 170-171, 183, 204, 224, 278

Scala MultiMedia INDEX

<i>DSS</i>	7, 74, 116-117, 132, 161, 213, 265, 282
<i>Edit menu</i>	13, 19, 24, 27, 29-30, 44, 47, 50, 119, 121-123, 136, 150, 158-168, 172, 176, 178-179, 182, 184-189, 209
<i>EX</i>	6, 14, 22, 73, 90, 109, 119, 126, 132, 13 155-156, 203, 220, 240, 278-281, 284-285, 288-295
<i>Execute menu</i>	119, 206-207, 219, 224
<i>File list</i>	40, 140, 147-148, 150
<i>File menu</i>	23, 37, 39, 57, 68, 106-107, 119, 123, 136-140, 146-151, 165, 174-175, 207, 2
<i>Font menu</i>	25, 41-42, 119, 163, 168-171
<i>Fonts</i>	4, 8, 10-11, 14, 19, 25, 48, 141-142, 169-171, 179, 182
<i>Fountain</i>	169-171
<i>Grid size</i>	179, 246
<i>Highlight</i>	6, 31, 85, 87, 150, 162, 196-198, 200, 2 251-252, 261
<i>IFF</i>	4, 8, 13-14, 29, 35, 46, 49, 63, 95, 101-112, 116-117, 128-129, 132, 136, 152-153, 158, 162, 166, 169-172, 180-186, 202, 205, 207, 213-214, 218-220, 223, 227, 229, 232-233, 254, 256, 265
<i>Inactive</i>	125, 162, 230, 235
<i>Install</i>	2-3, 6-7, 10-14, 16, 21, 107, 132-133, 2
<i>Interlace</i>	107-108, 231
<i>Joystick</i>	3, 9, 154, 200, 221, 233-234, 242, 248-249, 251-253
<i>Laserdisc</i>	279, 282, 286
<i>Layout menu</i>	45, 80, 119, 161-166, 170-171, 173, 176-183
<i>Left mouse button</i>	10, 16, 19, 27, 53, 83-84, 90, 121, 124, 131, 133, 159-160, 163, 199, 279
<i>Line spacing</i>	45, 179, 266
<i>Lingo</i>	7, 14, 119, 207, 210, 218-278, 284, 288 292, 295
<i>List menu</i>	47, 119, 121, 162, 184-186, 188-189, 192-195
<i>List view</i>	54, 63, 127-128, 135, 154

Scala MultiMedia INDEX

<i>Load menu</i>	165, 183
<i>Margins</i>	163, 166, 176, 181-182, 250, 273
<i>Menu colors</i>	153
<i>MIDI</i>	6-7, 116, 197, 240, 279, 294-295
<i>Multiple selection</i>	32, 160, 184
<i>Multitasking</i>	103, 116-117, 144
<i>Outline</i>	29, 45, 161, 173, 179, 199, 230, 235, 266
<i>Page commands</i>	136, 138
<i>Pause menu</i>	36, 100, 130-131
<i>Quit</i>	2, 20, 33, 38, 110-111, 113, 143, 225, 258
<i>Remap</i>	170-171, 179-180, 230
<i>Right mouse button</i>	24, 27, 34-35, 41, 44, 121, 124, 131, 158, 163, 193, 224
<i>Run!</i>	34, 58, 60, 64, 139
<i>Sampler</i>	116-117, 294
<i>Save default</i>	182
<i>Save IFF</i>	166
<i>Save layout</i>	182-183
<i>Save Script</i>	37, 140, 142
<i>Scala Key</i>	3, 8-9, 16, 142
<i>Scalable fonts</i>	170-171
<i>Scanner</i>	109-111
<i>Script</i>	3-8, 12-23, 30-31, 34-39, 46, 52-54, 56, 58, 60, 63-73, 78-79, 83, 88, 91, 93, 100-101, 108, 114-116, 121-143, 149-154, 156, 169, 181, 190-199, 206-212, 216-273, 280-283, 287
<i>Script commands</i>	138
<i>Set variable</i>	88, 198-199, 208
<i>Shadow</i>	29, 44-45, 76-77, 123, 160-161, 163, 165, 176, 180-181, 223, 230, 235, 266
<i>Shadow length</i>	45, 180
<i>Show</i>	4-8, 18-25, 31, 35, 37, 50-54, 58-59, 64, 67, 69, 73, 84, 87, 103-107, 112, 125-131, 135-139, 142, 148, 150, 153, 155, 157, 159, 162-164, 169, 174-175, 178, 180-227, 232-234, 239, 245, 250, 255, 257, 260, 262, 264, 269, 282-284, 286-288, 291-292
<i>Shuffler</i>	6, 21, 54-55, 63, 127, 135-136, 154

Scala MultiMedia INDEX

<i>Slider</i>	19, 22, 25, 40, 45, 124-125, 128, 130, 135, 147, 155, 172, 213, 215
<i>SMUS</i>	116-117, 132, 213, 265
<i>Snapload</i>	7
<i>Sound</i>	4-8, 13, 15, 18, 21-22, 41, 48, 67-71, 101, 103-104, 116-117, 119, 126, 128, 131-132, 135, 141, 150, 156, 164, 196, 212-218, 220-221, 223, 265, 280, 294
<i>Soundtracker</i>	7, 13, 69, 116-117, 132, 213, 265
<i>Static Page buffer</i>	
<i>System Information</i>	157
<i>System Menu</i>	7, 90, 119, 123, 127, 132, 135, 144, 152-157, 200, 221
<i>Text style</i>	122, 164, 230, 266
<i>Text wipe menu</i>	46, 61, 119, 184-189, 193-194
<i>The team</i>	157
<i>Touch screens</i>	201
<i>Variables</i>	6, 15, 18, 73, 75, 88-91, 93, 95, 101, 119, 199, 208-211, 218, 220, 247, 262
<i>Video digitizer</i>	109, 113
<i>Pause menu</i>	36, 100, 130-131
<i>Word wrap</i>	166, 181